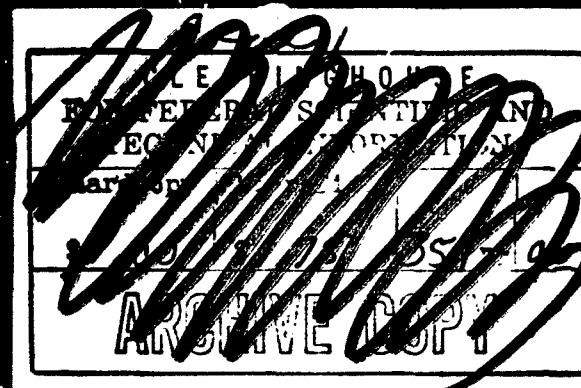


AD625417



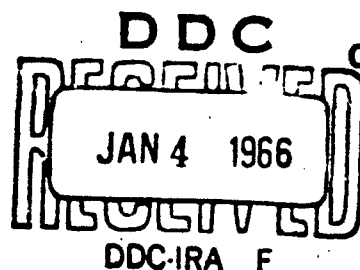
20050222058

TM-2624/100/00

PROCEEDINGS OF THE

SECOND SYMPOSIUM ON

COMPUTER-CENTERED DATA BASE SYSTEMS



1 December 1965

Best Available Copy

# TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of U.S. Government Contracts

PROCEEDINGS OF THE  
SECOND SYMPOSIUM ON  
COMPUTER-CENTERED DATA BASE SYSTEMS

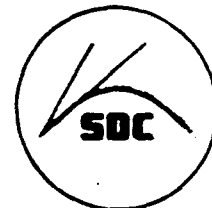
C. Baum and L. Gorsuch, Editors

1 December 1965

SYSTEM  
DEVELOPMENT  
CORPORATION  
2500 COLORADO AVE.  
SANTA MONICA  
CALIFORNIA

The views, conclusions or recommendations expressed in this document do not necessarily reflect the official views or policies of agencies of the United States Government.

Permission to quote from this document or to reproduce it, wholly or in part, should be obtained in advance from the System Development Corporation, or from authorized agencies of the U.S. Government.



ABSTRACT

The Second Symposium on Computer-Centered Data Base Systems was held on 20-21 September 1965. The symposium was sponsored by System Development Corporation, the Advanced Research Projects Agency, and the Electronic Systems Division, Air Force Systems Command.

After introductory speeches by spokesmen for the sponsoring organizations, two tutorial papers were given--a state-of-the-art survey of data base systems and a hardware survey. This was followed by individual work group sessions.

Prior to the symposium, a data base problem had been submitted to the designers of several data base systems. At the symposium, the spokesmen for these systems described how each handled the problem.

At the dinner, Captain C. W. Turner, DDR&E, spoke on computer-centered data base systems in support of high military command.

The work groups continued on the morning of the second day. Following lunch the participants saw laboratory demonstrations of several of SDC's data base systems--LUCID, General Purpose Display System, and ECCO-EPIC. Following the demonstrations, the chairmen of the work groups presented the results of their meetings.

TABLE OF CONTENTS

INTRODUCTION.....	v
SECTION I: INTRODUCTORY REMARKS.....	1-1
Donald L. Drukey, SDC.....	1-3
Robert W. Taylor, ARPA.....	1-5
Paul G. Galentine, Jr., Col., ESD.....	1-7
SECTION II: TUTORIAL PAPERS.....	2-1
State-of-the-Art Survey of Data Base Systems.....	2-3
The Impact of Hardware Developments on Data Base Systems.....	2-11
Computer-Centered Data Base Systems in Support of High Military Command...	2-29
SECTION III: FIVE APPROACHES TO THE SAME DATA BASE PROBLEM.....	3-1
Introduction.....	3-3
Description of the Data Base Problem.....	3-5
Three COLINGO-like Approaches to the Data Base Problem.....	3-31
COLINGO D.....	3-32
COLINGO C-10.....	3-59
ADAM.....	3-87
Mark III File Management System.....	3-123
On-Line Data Management System for the Massachusetts General Hospital.....	3-143
BEST System.....	3-185
Integrated Data Store.....	3-231
SECTION IV: DESCRIPTION OF SDC DATA BASE SYSTEMS	
DEMONSTRATED AT SYMPOSIUM.....	4-1
LUCID.....	4-3
General Purpose Display System.....	4-7
ECCO and EPIC.....	4-9



SECTION V: REPORTS OF THE WORK GROUP CHAIRMEN.....	5-1
Criteria for Going On-line.....	5-3
Entry and Query Language Design.....	5-7
File Organization.....	5-9
File Sharing and Protection.....	5-13
Theory of Data Base Problem Definition.....	5-19
Criteria for Evaluating Data Management Systems.....	5-23
Recording for Analysis, Costing, and Control.....	5-27

### INTRODUCTION

The Second Symposium on Computer-Centered Data Base Systems was held in Santa Monica, California, on September 20-21, 1965. Sponsored by the System Development Corporation, the Advanced Research Projects Agency, and the Electronic Systems Division, Air Force Systems Command, the meeting built upon the groundwork laid by the first symposium, held at SDC in June, 1963.

The important questions of how to organize the data stored in a computer, with the attendant problems of data generation, maintenance, and access, were illuminated through four different aspects of the two-day program: 1) tutorial papers on hardware, software, and user needs as related to data base systems; 2) a session in which the spokesmen for four operational data base systems illustrated how each system handled the same data base problem; 3) seven work groups dealing with various aspects of data management; and 4) laboratory demonstrations of three data base systems developed by SDC.

The meeting was attended by 185 persons representing 65 different organizations, including the various military services, federal and state governments, universities, industry, software firms, hardware manufacturers, nonprofits, consulting firms, and commercial institutions. This diversity of backgrounds seemed to have a healthy effect on the meeting, particularly in the work groups, where the variety of viewpoints served to stimulate each group into making a number of insightful observations and positive recommendations.

All aspects of the program are recorded in these proceedings. Of particular interest for the designer of data base systems are the detailed accounts of how the four operating systems (joined by a fifth in these proceedings) approached and solved the benchmark problem. The time and effort expended by the participants in this session, as well as of all the others associated with the symposium, are gratefully acknowledged.

Claude Baum  
Program Coordinator

SECOND SYMPOSIUM ON COMPUTER-CENTERED DATA BASE SYSTEMS

sponsored by

System Development Corporation  
Advanced Research Projects Agency, Department of Defense  
Electronic Systems Division, Air Force Systems Command

First Day: Monday, September 20, 1965  
Santa Monica Civic Auditorium

- 8:30 Coffee
- 9:00 Welcome and Opening Remarks - Dr. Donald L. Drukey, Symposium Chairman  
Manager, Research & Technology Division, SDC  
Robert W. Taylor, Deputy Dir., Information Processing Techniques, ARPA  
Col. Paul G. Galentine, Jr., Deputy for Engineering & Technology, ESD
- 9:15 State-of-the-Art Survey of Data Base Systems - Guy H. Dobbs, Manager, Computer Center Dept., SDC
- 0:00 Coffee
- 0:15 The Impact of Hardware Developments on Data Base Systems - L. C. Hobbs, President,  
Hobbs Associates, Inc.
- 1:00 Concurrent Work Groups - First Meeting
1. Criteria for going on-line - Chairman: Ralph G. Tuttle, ONR
  - 2., 3. (Deleted)
  4. Entry and query language design - Chairman: Dr. Frederick B. Thompson, Cal Tech
  5. File organization - Chairman: Jack F. Nolan, M.I.T.
  6. File sharing and protection - Chairman: Richard G. Canning, Canning Publications
  7. Theory of data base problem definition - Chairman: John W. Young, Jr., NCR
  8. Criteria for evaluating data management systems - Chairman: Herbert Koller, U.S. Patent Ofc.
  9. Recording for analysis, costing, and control - Chairman: Col. Paul G. Galentine, Jr., ESD
- 2:00 Catered Lunch at Santa Monica Civic Auditorium
- 1:10 Four Approaches to the Same Data Base Problem
- Introduction to Session - Richard G. Canning, President, Canning Publications
- 1:20 COLINGO - Frank Cataldo, Technical Staff, The MITRE Corporation
- 2:10 Mark III System - John A. Postley, Director, Advanced Information Systems, Informatics, Inc.
- 3:00 Coffee
- 3:10 On-Line Data Management System for Massachusetts General Hospital - Paul Castleman, Director,  
Client Service Group, Bolt Beranek & Newman Inc.
- 4:00 BEST System - James R. Ziegler, Manager, Programming Services, Electronics Division, NCR
- 4:50 Summary of Session - Richard G. Canning
- 5:00 Adjourn
- 7:00 Social Hour at Santa Ynez Inn
- 8:00 Dinner at Santa Ynez Inn
- Speaker: Captain Charles W. Turner, Military Assistant, Director of Defense  
Research and Engineering

Second Day: Tuesday, September 21, 1965  
Santa Monica Civic Auditorium

- 8:30 Coffee
- 8:45 Concurrent Work Groups - Second Meeting
- 1:45 Catered Lunch at Santa Monica Civic Auditorium
- 1:00 Introduction to Laboratory Demonstrations - Alfred H. Vorhaus, Head, Data Base Systems Staff, SDC
- 1:40 Adjourn to SDC
- 2:00 Laboratory Demonstrations of SDC Data Base Systems: LUCID, General Purpose Display System, ECCO-EPI
- 3:30 Reports of Work Group Chairmen (2500 Commons Room, SDC)
- 4:50 Concluding Remarks - Dr. Donald L. Drukey, SDC
- 5:00 Adjourn

Place: Santa Monica Civic Auditorium, 1855 Main Street, Santa Monica, California  
and  
System Development Corporation, 2500 Colorado Avenue, Santa Monica, California

SECTION IINTRODUCTORY REMARKS

The theme of the symposium--namely that there has been considerable progress in data base technology but that much more needs to be done--was voiced in the introductory remarks of the representatives from the three sponsoring organizations: System Development Corporation; Advanced Research Projects Agency, Department of Defense; and Electronic Systems Division, Air Force Systems Command.

ARPA and SDC had cosponsored the first data base symposium in June 1963. Both ARPA and ESD have been vigorous supporters of research and development in computer-centered data base technology at SDC and elsewhere.

Donald L. Drukey, SDC.....1-3

Robert W. Taylor, ARPA.....1-5

Col. Paul G. Galentine, Jr., ESD.....1-7

Opening Remarks: Dr. Donald L. Drukey, Symposium Chairman  
Manager, Research and Technology Division,  
System Development Corporation

I would like to welcome you all on behalf of SDC. We here have a considerable commitment to data base problems. We are concerned with the fact that data bases and the handling of them by people is one of the dominant problems of our day, particularly in the military area. For that reason, we are convinced of the importance of this problem. This symposium is being held in the hopes that we will communicate with one another and perhaps even learn or generate something new. The program we have set up is aimed fairly broadly at the problem of handling large files of formatted data.

I hope that the groups will be particularly concerned with the area where I believe the military and the management problems lie. This is in the use of large files for ill-defined purposes. Most data base problems are characterized by the fact that we don't really know what belongs in the file in advance, how it should be organized, or how we will want to retrieve information from it. Moreover, as the managers and our military command personnel work with the data in their files, the things that they want to do with the files change.

In the past we have all dealt with large systems that are aimed at handling voluminous files. Payroll-recording personnel and those engaged in insurance tasks have presumably solved this problem quite well because they knew in advance what they wanted those data for and how they wanted to use them.

On the other hand, as we tried to apply similar techniques to military and management problems, we have not done as well. The reason for our difficulty is that the tasks of the command and control personnel and management personnel are poorly defined. A manager, for instance, often cannot predict the form in which he will require information, what kind of access he will need, or how he may want to interact with it. The only way to investigate all the possible factors may well be to allow our systems to change as use and experience indicate they should change. This speculation is aimed at that part of the data base problem that I believe is most germane to the purpose of this symposium. I hope that all of you in the working and the discussion groups will address yourselves to these relatively unexplored areas as well as to the more familiar ones.

The problems associated with allowing our system to change to reflect the necessities of use and experience are far more significant than problems associated with canned, static situations where the task consists of simply inserting the data in the files, keeping it up to date, and producing the same old reports from it. I do not think that we here are as much interested in the latter problem as we are in the former. I am firmly convinced that, as we learn better how to solve the more difficult problem, we will find that there are many situations in which our data will be used empirically; that we

1 December 1965

1-4

TM-2624/100/00

will be experimenting with the things that can be deduced from the data. I believe that both intelligence and operational activities will require a solution to this problem, and I know that managers will require an answer, too. Very typically, managers massage the data or have it massaged for them for some reason which is not clear in their minds. Ultimately, it is the data themselves which suggest to them the answers they are looking for.

Opening Remarks: Robert W. Taylor  
Deputy Director for Information Processing Techniques  
Advanced Research Projects Agency

On behalf of ARPA as a cosponsor to this symposium, I'd like to welcome you. ARPA indeed has an interest in learning how to handle large masses of data in an interactive way. We are supporting work in this area--here at SDC, within Project MAC at MIT, and in other places where our research contracts are carried out. However, we are interested in the data base problem from a particular point of view. I might best describe this in an over-simplified form by referencing an argument that I've heard raging from time to time. There are a group of people who say that the data base and the naive user should really not come together. That is, they say that data base loading, organizing, and reorganizing operations should be kept away from the naive user. The reasoning is that in doing that kind of reorganization, a naive user will probably obliterate his data base to the point of uselessness. The converse point of view says that the naive user is the man who knows best about what is, after all, his data base. And, in fact, the software will have to be organized so that naive users can indeed load and reorganize their own data bases. It's with this latter point of view that I, representing ARPA, would like to identify myself and our program. I think that people who speak about keeping naive users, owners, and proprietors of data bases away from the ability to organize and reorganize their data bases don't understand the problem.

Let me also mention something about the querying aspects of handling large data bases. A number of people in the Department of Defense and elsewhere have come to ARPA and asked what we were doing in the area of natural language research. When we reference some restricted English efforts, some of our visitors have correctly told us, "Yes, but that's too restricted. I can't learn another language." To date, so-called "natural language" systems may well read like natural language in computer output but these systems don't write so naturally; for example, a word such as "greater" used in a query could not be replaced by "exceeds." In short, it may be impossible to build a computer program that can be generalized completely.

In order to suggest a way out of this difficulty, I'd like to distinguish between a restricted language and a language that will come back to the user in the case of an ambivalent query. Perhaps it will be easier to build a query system which will respond conversationally to an ambivalent query than to build one that will be written in natural English. For example, there's a phrase that the people in Los Angeles use in connection with their weather. The phrase is "light eye irritation." Perhaps it is more reasonable to expect a computer program to respond with, "Are you most concerned with optics, opthamology, eye color, or atmosphere?" than it is to expect the computer to get the correct meaning from context.

Opening Remarks: Colonel Paul G. Galentine, Jr.  
Deputy for Engineering and Technology  
Electronic Systems Division

On behalf of the Electronic Systems Division of the Air Force Systems Command, United States Air Force, I'd like to welcome you to the Second Symposium on Computer-Centered Data Base Systems.

I believe that a common approach used in making introductory remarks at a gathering of this kind is to begin with a survey of past accomplishments, followed by a look at the current state of the art. Not infrequently these comments are mingled with a hint of backslapping by the speaker.

Upon looking at the current state of affairs, and upon viewing the magnitude of the problems still facing us, I am somehow not inclined to indulge in backslapping. A technology which, every day, is seeing demand pull further ahead of resources has little room for self-approbation. I think it more appropriate to commence with a look at the problem; to consider the challenge; and, if possible, to indicate the direction where a solution may lie.

If our past experiences have taught us anything, it's that we need a change in approach rather than a mere refinement of our present techniques. In the design and production of a software system, we stand today at a point analogous to the point where hardware production stood two hundred years ago--in the days prior to the industrial revolution. At that time all production was dependent on the efforts of a limited number of skilled craftsmen: men who had spent years in apprenticeship, acquiring proficiency in their trades. Until the process of producing goods was freed from the output of the artisans, supply was doomed to fall further and further behind the demand of an expanding population.

Today the demand for new data management systems of every description is increasing at a tremendous rate. By and large, computer hardware demand has been responsive to the demand and has evolved rapidly, permitting an expanded number of different applications. In my opinion, however, conventional programming techniques, although now refined and sophisticated in many respects, can not hope to keep pace with requirements.

What is needed today is a sort of industrial revolution in software production. We can no longer afford to be limited by the laborious manual efforts of a few software artisans. Rather, tools must be provided which will permit the semi-skilled, nonprogramming specialist to produce software systems.

In one sense, the parallel I have drawn with the industrial revolution is a poor one, because it masks complexities of the system design process which really have no exact parallel or precedent. The customer of two hundred years ago had only to say to the craftsman, "I want a cabinet," or "a pair of shoes." The precise nature of the user's requirement was clear; production to meet the requirement was the problem. Today the data system user, particularly in the



Air Force, must communicate his needs to a system acquisition agency, which in turn communicates them to a development agency or commercial concern. It can become difficult for the user to communicate his requirements, even if he knows precisely what his requirements were (which frequently, of course, he doesn't). The system design process is commonly a user education process as well, with the user becoming increasingly aware of his exact needs and increasingly capable of specifying them as the design process progresses. Unfortunately, the system design is, at the same time, becoming increasingly less capable of accommodating these user requirements.

As tedious, expensive, and time consuming as the design, production, debugging, installation, and checkout process is, the user has yet further serious problems once the system is installed and operating. In anything as complex as a large data management system, it is only through use that the customer discovers its limitations and deficiencies, and it is only after debugging that new requirements and new applications become apparent. It is at this point that customers frequently become dissatisfied with their new system, for they find that system changes, even changes that appear trivial, can be as expensive and time consuming to accomplish as the original system was to produce. Indeed, large systems, over their life cycles, have frequently incurred costs for changes and updates which are several multiples of the original system cost.

If there is a solution to the problems mentioned above, and I think there is, it would seem to lie in the direction of equipping the user to participate more easily and actively in his own system design; and, once the system has been implemented, to modify that design more easily to meet changing requirements. This approach (one that we might refer to, in an oversimplified way, as the generalized software approach) is at last receiving a considerable amount of attention, and much is being done or is proposed to be done in this direction.

Without being unduly pessimistic, I nonetheless feel that neither is enough being done nor is the rate rapid enough to meet the challenge. I hope that this symposium will bring forth many items for discussion which will aid the user to play an increasingly important role in data system design. It is up to us to provide him with the tools necessary to do the job.

SECTION IITUTORIAL PAPERS

To enable the attendees of widely diverse backgrounds and experience to participate on a more-or-less equal footing, several tutorial papers were presented. Since the papers had to be of reasonable size, the data base field was arbitrarily partitioned into three aspects: the software, the hardware, and the user. It was recognized that these three topics were interdependent, and each speaker attempted to consider the over-all data base problem, but from a different viewpoint.

State-of-the-Art Survey of Data Base Systems.....	2-3
The Impact of Hardware Developments on Data Base Systems.....	2-11
Computer-Centered Data Base Systems in Support of High Military Command.....	2-29

## STATE-OF-THE-ART SURVEY OF DATA BASE SYSTEMS

Presented by: Guy H. Dobbs  
Manager, Computer Center Department  
System Development Corporation

In thinking about this presentation, I realized that what people in our field generally mean by "state of the art" is a combination of four parts theory and one part practice. In our profession we often talk about things that we would like to do or that we have demonstrated in the laboratory, but that have really never been put into an operational or production environment. In view of the kinds of systems we envision, I believe that the relative lack of progress in developing operational data base systems is appalling.

The comments that I will make this morning are going to be concerned with a fairly limited area, compared to the totality of what data base systems could include. My talk will be confined to structured data files and the means for generating, maintaining, and accessing them. I will not talk about text or library systems, although they involve many of the same problems and techniques. I will discuss structured data files from the perspective of the capabilities that are being used or have been demonstrated in an operational system. In mentioning various characteristics, features, and design points, I will, unfortunately, not have time to give credit to many of those responsible for them.

Before surveying the kind of capability that is offered by existing systems, it is important to try to understand by whom the system will be used. We can understand who the user is if we try to determine the fundamental objectives of contemporary data management systems. At least four different objectives can be stated:

- . The first objective is that data base systems should provide information processing support to decision-making and intellectually creative processes. These data base systems support management information systems, command control systems, basic research, etc., and their users are executives, commanders, and researchers. Such systems are used to manipulate information so that the user can find out what the problem really is.
- . The second objective is to reduce system implementation time and cost, in particular, programming, reprogramming, and conversion costs. Many producers and users of these systems admit that the systems are aimed at a programming user. Examples of such systems are compilers, sophisticated support and operating systems, generalized file and data reduction systems, public program libraries, etc. The users are systems analysts and applications and systems programmers; the language and techniques used in these systems are from the programming domain.

- A third objective of contemporary data base systems is to decrease problem definition time and cost. In this case, the applications are for information and data processing system design, as is done with ADAM in the Electronic Systems Division system design laboratory, or for engineering design automation, or for social science research. The users in this case are content specialists and professionals in some particular domain.
- The fourth objective is to increase user control over the creation and maintenance of, and access to, data. The terms in these systems are generally user-oriented, reflecting that the user is not a programmer. Typically, these kinds of users turn out to be clerks, technicians, staff people, and operators, who support the objective of providing information processing capability for decision-making and creative processes.

Probably no systems exist today that interface directly with the executive user. As a matter of fact, the majority of data base and file-handling systems operating today are used by and aimed at supporting computer programmers. To sum up, executives, commanders, and researchers are interested in finding out what the problem is; information processing system designers use the systems to solve particular problems; computer programmers largely implement initial problem solutions; and clerks, technicians, staff, and operators keep those solutions going or those systems operating. Therefore, I believe it is a fallacy to talk about a single user. Software of varying types must be provided to correspond to the differences in individual users and tasks, because no single computer program system can serve the variety of objectives and users that I have noted.

Despite the differences in objectives noted above, there is substantial agreement among the various users about which functions such systems should support. There are five major functional areas in which software support is needed and provided. One area is that of data base definition. It includes data description, structure and organization; converting this data description into usable machine-processable form; and accomplishing other data management functions. This area, in effect, is a language, and most of the language we see used today in data base and data management systems is a programmer language, even in cases where a fairly sophisticated query capability aimed at nonprogrammers exists. These languages are deeply rooted in procedure-oriented techniques, to which they form a minor extension. A second capability that is needed is one for creating the file or data base--i.e., for inserting and storing data values; in short, the input process. Next, a maintenance capability is needed for a data base, enabling the user to alter structures, data values, organizations, definitions, or procedures. The fourth capability is that of retrieval: some means for accessing data or selecting the kind of data that are needed. The last capability calls for output processes to support any kind of user; it includes techniques for presentation, reporting, and displaying data.

Now, if I may separate the two major classes of users into programmers and all others, it seems to me that two fundamental points of view emerge in viewing today's data base systems. One of them is an internal point of view, concerned with the machine, machine word allocation, devices, and computer programs; it is quite clearly oriented to computer programmers. As nearly as I can tell, the vast majority of the systems that we have today are described that way and are used in that fashion. On the other hand, for all other users, there is an external point of view. This kind of user thinks about data, about the information processing functions, about his problem, his task, his operation. It is very important to remember this distinction because we often fall into the trap of bandying words without realizing that they mean entirely different things to different people. For example, when IBM talks about data management in connection with its new 360 system, it is talking about the former use, about methods of storing data in machine form and machine methods of accessing data on specific devices. Data management, in this sense, is not concerned with user-oriented functions, except as that user is identified as a computer programmer.

Compounding this semantic confusion is the added problem of a nomenclature in constant change. Yesterday's file is today's data base; today's data base will be called a data set tomorrow. These data sets, files, data bases, or whatever you wish to call them, will contain not only the static kinds of information that we are used to processing, but entire computer programs, and this is as it should be. Therefore, there exists today a concept of two kinds of files--the kind that we are normally familiar with, and the so-called executable file, which consists of a computer program. Furthermore, physical devices and media that we used to refer to as physical files (that is, tape, disc, and card files, etc.) have a new nomenclature: they will be called volumes. A volume can be a tape, a disc, a tapestrip, etc. What we used to call records are now called entries, and some people propose to call them lines in the future. What we used to call a block, which was really a physical allocation of data, will be called a page. Finally, many new terms are being added, such as segments which identify logical entities. It is very hard to keep up with the game when you can't keep up with the names.

Let's look in a little more detail at some of the particular functions that I have mentioned and attempt to describe briefly some of the capabilities that are found in contemporary data base systems.

First, in terms of definition and description capability, the most widely used level of data description capability is one that allows the user to give a data element a name and a symbolic class or category and a symbolic encoding label. Such techniques are commonly used in COBOL, and (with a good deal more detail) in "Compool" oriented data definitions, and in some form by almost all program systems. There have been attempts, notably at MITRE and at SDC, to provide a user-oriented language as opposed to a programmer-oriented language, to allow a user to describe his data directly to the computer and to provide

supporting software to turn this data description into a machine processable form. This problem is not so difficult, and it is hard to understand why such capability does not exist on a wider front. Perhaps the reason for it is that, having put a considerable investment into existing file structures and existing techniques, we are loath to change. Sometimes new generations of hardware help to force this kind of change.

Very few operational systems have the capability to describe complex structures directly in the description language, although some laboratory systems do have it. Yet, one wants to be able to describe conditional data elements, string or repeating groups of elements, etc. In almost all operational systems, the typical unit that the data represents is carried separately as semantic content, that is, either external to the system or embedded in code which is external to the data description. For example, very few systems enable the user to define units of measure and to use these units in the conversion of data or in subsequent processing. On the other hand, a good many systems do provide capability for the user to use synonyms and noise words in describing his data, and some systems are beginning to provide the capability to insert delimiting and range statements which express the limits or the range of the values for potential data. It is certainly within the capabilities of present technology to provide a symbolic language with which an unsophisticated user can describe fairly complex data structures, and can get automated aid in handling the details of machine allocation, machine packing, and machine interpretation. This is not yet done in general practice.

A second badly needed capability is one for describing more completely the data base organization and structure, including class membership, set membership, ordering criteria, sequencing criteria, subsetting criteria, linkage criteria, association, etc. Historically, most files have been designed on the premise that one knew how one wanted to access them in the future. However, a characteristic of emerging large data base problems is that one does not know how one wants to organize the data. This means that all data names or all data values, a priori, have equal weight as an accessing key. Although this approach appears to be sensible, it poses a rather horrible problem for the designer of generalized software. Therefore, if a user can specify certain kinds of criteria for organizing as he learns more about his data, many of the problems of supporting generalized software can be solved.

In connection with data base organization and structure, I notice that most systems in operation today are based on serial file techniques. Although for the last five years we have heard a good deal about the obvious advantages of the more powerful list structures, in practice list structures have found limited usage. However, those applications that have been made are quite advanced; no longer does one find a simple list with an element that tells you where to find the next value or the next entry. Instead, the searcher finds backward and forward links, up and down links, cross links, inverted links, circular links--in short, the list processing designers have developed

extremely sophisticated structures for handling sophisticated problems. However, implementing these structures for most operational systems is difficult, because one runs into the problems of space and efficiency. To date, the appropriate kind of hardware for dealing effectively with list structure has not been available and this has probably impeded progress in this area. Where list structure has been used effectively, the efforts have been primarily in terms of adding the capability to an operating system for dynamic storage allocation of the data base itself.

In the area of file organization and description, an attempt has been made in recent years to keep the data description separate from the data file itself. This practice is going to be reversed (or is in the process of being reversed) in some contemporary systems and in future systems. When we talk about some advanced software in a few minutes, the reason for this reversal will become more obvious. In any event, there is increasingly heavy utilization of self-defining files, in which the file itself contains all of the information that is needed to describe it. This feature can turn out to be an extremely powerful capability, provided file users agree upon a standard method for at least that portion of the file description. Many people, however, continue to maintain file independence by utilizing dictionaries entirely separate and external to the files themselves. This practice has certain disadvantages in that, in most cases, it is not possible to use the same generalized system to process the dictionaries as is used to process the files. Ideally, it should be possible to use the same system to process both.

Much of the rationale for using the existing file organizations has come from the heavy utilization of serial storage devices, that is, tapes. Even in the presence of high-speed, random-access equipment (i.e., disc files) it is not clear that the most effective processing strategy is to randomize the file in every case. Many users are finding that the nature of many data base processing operations is such that 90% of the time the file will be processed serially; if one deliberately randomizes the file to take advantage of random-access storage, he may find that the search for information takes longer than it did with the tape file. Therefore, people are beginning to take a second and a closer look at the use of random-access files and how data ought to be organized for them. One technique which seems to be useful here is the so-called inverted file, sometimes called a concordance, where entry to the file is made by value. This technique is based upon having all existing and potential values in the file maintained as entry keys. From these, one derives an address and subsequently can find an entry. One criticism of this kind of file organization is that it is extremely wasteful of space; in some instances, the inverted file requires twice as much storage space as does the original data base. However, in cases where the file is extremely large and contains a large amount of redundant data values, these files may be practical.

Let me briefly review the types of contemporary file organizations. We have mentioned that serial files are predominant; there are things called parallel files which are really another form of serial files. Change file lists, as I pointed out, are now knotted, threaded, back-linked, forward-linked, etc.; and there are things called rolls, dictionaries, directories, concordances, and compools, all of which allow access to data, or enable one to describe the data or the data organization.

What about the available maintenance capability? Most systems contain good capability for data base maintenance these days, albeit handled by computer programmers. Most systems have a powerful capability to link data elements, to add new data elements, or to delete entire segments or portions of files. However, there is very little capability to change data descriptions, file descriptions, or file organizations. In most file systems that are not supported by generalized software, users have realized that their files are obsolete, but the costs of new software required to change the files are prohibitive.

One serious problem in file maintenance is the lack of ability to share files among simultaneous users. Very few systems have solved this problem adequately. The converse of this is the security problem, i.e., how to restrict file sharing to authorized personnel; again this problem has proved to be difficult to solve in many contemporary systems.

Turning to the area of retrieval, many of the more general systems--in particular, the recent on-line systems--are developing query language aimed at making it easy to implement a selection or retrieval procedure. In this area, capability is being provided to ask quite sophisticated questions. For example, one may use conditional statements, or chain queries together, or save queries and subsequently compile them. In terms of the query language capability that is provided, the retrieval process bears a close resemblance to computer programming, except that it is at a slightly higher level. Many of the file organizations that I have described can, by virtue of the way in which they have been structured, facilitate or hamper this retrieval and selection process. For example, proponents of the inverted file and concordance file would submit that this method of organization gives equally rapid access to any place in the files on the basis of any kind of selection or criteria statement that one can think of. Proponents of other organizations would make similar claims for their particular structure to satisfy certain kinds of selections and retrieval requests.

Finally, there is the area of presentation. Although report generators have been with us for some time, it is surprising that their capabilities have not advanced over the last few years. In fact, I would say that there has been no significant improvement in report generation capability since the generators first appeared. Many users would like to have multi-dimensional rather than two-dimensional reporting capability. Most important, however, is the potential and emerging utilization of cathode ray tube displays.



Cathode ray tube displays are offering, for the first time, an intimate interaction between the user, his presentation, and his data. As has been mentioned before, problems to be addressed with these kinds of systems in the future will indeed be ill-structured, and the potential user will come to his supporting system with few preconceived or preplanned notions of how to interact with his data. The promise of display tubes and the appropriate supporting software is that they will allow the user to interact with his data in a much more meaningful fashion.

However, there are some dangers that are not only associated with the interaction of a user and his data, but are also relevant to our discussion of query languages. In devising programming languages, query languages, and display languages, we may have over-reacted to the early days when we coded in binary and octal, by attempting to make the newer languages formal "man languages." However, it is not clear if language used in man-machine dialogues should be "machine language" or "man language." What I am pointing out is that not nearly enough attention has been devoted to the problems associated with the use of entirely different languages, to the problems of context-dependent languages as opposed to context-free languages, the problems of informal languages as opposed to formal ones. If one reflects, one might find that often more information is conveyed by lack of precision--e.g., in the way in which we talk with each other--than by formal precision, and that it might be worthwhile to look for techniques and ways to exploit this phenomenon usefully in data base and information processing systems.

So much for the general characteristics that are found in contemporary data base and data management systems. You will note by the tenor of my remarks that I find no cause for real rejoicing. Let me, however, temper my remarks by pointing out some considerations that I believe are emerging, and that hold some promise for the future. I believe that in the future the impact of economically defensible, time-shared, on-line systems is going to be felt very strongly in data management. Their impact is already being felt in a number of ways. First, the designers are realizing that it is impossible to separate the problem of data management from that of executive control and resource allocation in the processor. That is, in an environment that must serve a multiplicity of users with diverse interests, and that must manage, from a machine point of view, their files for them, there must be an intimate relationship between executive control and data base management capability. Furthermore, if time-shared, large-scale, multi-access systems are to continue to exist or to be economically defensible for large numbers of people, they will eventually become memory-centered, large, file-based systems in which high-speed core will appear to most users to be a buffer and the user's real concern will be the large-scale belt memory. Furthermore, from the user's point of view, the machine can be considered to have an infinite amount of memory, freeing the user completely from the constraint of having to worry about any limitations.

There is reason to believe that recent hardware improvements and certain approaches to implementing executive system design and data management enable exploitation of just that kind of capability. For example, the data management capability envisioned in the IBM 360 series will eventually allow a user to look at a capability from a point of view of some 400 billion bytes of memory. If that isn't enough for him, heaven help him. Furthermore, this kind of capability will allow a user to look at the available data management capability from the viewpoint of symbolic names or symbolic areas for data. In addition, another important aspect is that the same procedures must be usable for processing both data files and executable files. The capability to use procedures in common means that the procedures must be managed in the same manner that any other data element or data set in such a system is managed. This means that the problem of sharing files and the problem of selective access to files must be solved, because clearly one cannot afford multiple copies for some of the common procedures needed by all users. For example, I believe that at last count in Project MAC there were about 5,000 routines in the public library, and this figure grows every day. Each of these routines is in the public library because each is presumably useful to more than a small number of users. If these routines cannot be shared, we will not be able to make very effective use of these kinds of systems.

In conclusion, I believe that recent hardware developments and advanced software technology are going to present us with the catalyst of necessity to force us to look at many of our problems associated with the data base in an entirely different fashion, to throw off some of the shackles of the serial files and of the way in which we have done business in the past. The problems that we are going to have to solve are those which are apparently developing faster than the technology is developing. If we do not or cannot find new kinds of approaches to some of them, we may never catch up.

## THE IMPACT OF HARDWARE DEVELOPMENTS ON DATA BASE SYSTEMS

Presented by: L. C. Hobbs  
Hobbs Associates, Inc.

### INTRODUCTION

Since this symposium is concerned with computer-centered data base systems, the considerations of hardware developments will be limited to storage and retrieval systems using digital computer technology. Only hardware for storing digital data will be discussed, although there are techniques for identifying and addressing microfilm documents under computer control. The kinds of equipment that are implied by the functional requirements for storage and retrieval in a data base system and the types of equipment in each category will be considered first. This discussion will be followed by a discussion of hardware capabilities anticipated by 1970 and the impact of new hardware technologies on each of the types of equipment and on the over-all data base system. Finally, some of the major problem areas facing the designers of future computer-centered data base systems will be listed.

During the course of a study we are making for the Office of Naval Research of the applications of technology for Tactical Data Systems in the 1970 to 1980 era, we have been impressed with the truly significant progress and changes that will be made during the next five years in some types of equipment in computer-centered systems and, also, by the relative lack of progress that will be made in other types of equipment. This is equally as true for data base systems as it is for tactical systems--perhaps to an even greater extent. We have been concerned by the lack of understanding of the impact and implications of these differing rates of progress on future systems by the majority of those in the field. Although there is a general awareness that progress appears to be faster in some areas than others, very little is being done about it. Realization of the tremendous potential offered to users of computers and data processing systems by the almost incredible progress in integrated circuits and other batch-fabrication technologies will be seriously limited by the system imbalance resulting from the lack of comparable progress in peripheral equipment, such as large capacity mass memories and input/output equipment. Some of these problems will be pointed out in this paper, and their impact on data base systems will be considered.

In a system for entering data into a data base, storing the data base, and retrieving data, the following functions are inherent:

- . Entering data into the system from the external world.
- . Positioning this data in the proper location in the data base.
- . Storing the data contained in the data base.
- . Requesting the data from the data base by the user.

---

Most of the material on future hardware technologies presented in this paper was developed in the course of a study sponsored by the Office of Naval Research. However, the opinions expressed in this paper are the responsibility of the author and do not necessarily represent the views of the Office of Naval Research.

- . Processing the request to determine the portion of the data base required and its location.
- . Retrieving the desired data from its location in the data base.
- . Presenting the requested data to the user.
- . Changing of data in the data base by the user.
- . Outputting data from the data base to some other medium, such as hard copy, when required.

Although this paper considers only hardware technology, obviously some of the requirements listed above have important software implications as well.

#### EQUIPMENTS FOR DATA BASE SYSTEMS

The hardware for implementing the functional requirements listed above can be conveniently grouped into four major categories of equipment as follows:

1. Input/output equipment for handling the functions of entering and outputting data.
2. Processor and internal memory for handling the requirements of positioning data, processing requests, and retrieving data.
3. External or auxiliary storage for storing the data base including both on-line and off-line types of mass memory and bulk storage.
4. Inquiry/display consoles for requesting data, presenting the results to the user, and changing data.

In some cases it is conceptually difficult to draw a clear-cut line separating particular pieces of equipment into one of these categories or the other. For example, magnetic tape, which is basically an off-line storage medium, is frequently considered input/output equipment. Certainly, some of the basic technologies applicable to on-line auxiliary storage or mass memories may also be applicable to internal memories.

Some of the major types of equipment in each of these categories include:

#### Input/Output Equipment

- Keyboards
- Character of print readers
- Magnetic tape
- Punched paper tape
- Punched cards
- Printers

Central Processor

Logical circuits for implementing control functions and arithmetic or logical operations  
Registers or high-speed control memory  
Main internal memory  
Associative memories

Storage

On-line solid-state mass memories  
On-line electromechanical mass memories  
Off-line bulk storage

Inquiry/Display Consoles

Electric typewriters  
Teletype units  
Keyboard/CRT inquiry stations  
Inquiry stations including small satellite computers or processors

The items listed under Central Processor above are the devices used in implementing the processor rather than individual types of equipment as in the other categories.

The present state-of-the-art in hardware can be indicated by the characteristics of a few selected items that might be used in a data base system:

Magnetic tape units - 20,000 to 200,000 characters per second;  
5 to 20 million characters per reel

Printers - 600 to 1500 lines per minute;  
120 to 130 characters per line

Logic circuits - 5 to 20 nanoseconds propagation delay per stage;  
clock rates of 1 to 10 megacycles

Magnetic core matrix memories - 1 to 6 microseconds cycle time;  
2,000 to 64,000 words capacity

Magnetic disc files - 20 to 200 milliseconds access time;  
40 to 250 million characters capacity per file

In most cases the characteristics listed above do not represent extremes but represent a range of typical values of equipment commercially available today.

## HARDWARE CAPABILITIES ANTICIPATED BY 1970

In analyzing and evaluating research and advance development efforts presently underway in different areas of hardware technology, during the ONR study referred to previously, it has been apparent that the most significant advances will be made in memory and logic components. Although there will be improvements in input/output equipment, minimizing the need for input/output in the conventional sense offers the best hope for over-all systems improvements. Integrated circuit technology will bring revolutionary changes in the size, cost, and reliability of logical components. Lesser improvements will be realized in circuit speed. Memory technology will provide significant improvements in speed, capacity, cost, reliability, and size.

The major improvements in displays will be in cost and in the determination and implementation of the proper functions from the user standpoint. The cathode-ray tube (CRT) will probably remain dominant as the visual transducer for console displays through 1970, but there are several new techniques under development that may eventually replace the CRT in many applications. The advances in memory and logic component technologies will permit significant improvements in the logic and memory portions of console displays.

In considering future capabilities of basic hardware technologies, it is helpful to use a different grouping than the four equipment categories discussed previously, since basic logical components and storage techniques are used in portions of the system (e.g., displays) other than central processors and storage units. Advances anticipated in input/output equipment, logical components, storage, and displays will be discussed in greater detail.

### Input/Output Equipment

There are three major approaches to improving the performance of future systems with respect to input/output equipment. These are:

- . Improvements in the performance of present types of input/output equipment.
- . Development of new types of input/output equipment that are not in widespread use at present.
- . System organization approaches that minimize the need for conventional input/output equipment.

Each of these approaches will play a part in performance improvements in future systems. However, unless much greater effort is placed upon the development of non-mechanical input/output equipment, the best hope for future systems probably lies in developing system techniques that minimize the need for input/output equipment.

Almost all present types of input/output equipment are electromechanical. This imposes limitations on the improvements that can be achieved and on the ability to utilize the benefits of batch-fabrication of electronic and magnetic components. Although these electromechanical input/output equipments will limit systems performance, the effect on systems cost and reliability is even more serious. The performance limitations could be overcome to some extent by using a larger number of input/output units, but use of more units further accentuates the cost and reliability imbalance with respect to the central processor and memory.

Performance characteristics anticipated by 1970 for some of the principle types of conventional input/output equipment are shown in Table 1. Examination of these characteristics will indicate performance improvements of less than one order of magnitude (and in most cases of less than two-to-one) over equipment commercially available today. For example, speeds available today of 200,000 characters per second for magnetic tape units and 1,500 lines per minute for impact type printers were cited previously.

Punched paper tape is not included in Table 1 because it is believed that incremental magnetic tape readers and recorders will replace punched paper tape equipment for most high performance applications. Incremental magnetic tape equipment will be cheaper for high performance, will be more reliable, and will utilize tape records and formats that are completely compatible with high speed conventional magnetic tape units.

Several new types of input/output equipment are under development that offer promise for performance improvements in future systems. These include:

- . Character recognition and print readers
- . Voice recognition and voice output
- . Non-mechanical keyboards
- . Graphic input
- . Solid-state replacements for magnetic tape equipment.

Some of these, such as optical character readers, are in limited use at present; others, such as voice recognition equipment, are probably ten years or more away.

The term "character recognition" is applied to a broad range of devices from relatively simple ones capable of reading controlled and highly stylized magnetic ink printing on bank checks to ones capable of reading fifteen or twenty different type fonts on pages of printed documents. By 1970, equipment capable of reading 2000 to 3000 characters per second from a printed page should be available. Advances in integrated circuit logic components and memories discussed later will provide significant reductions in cost because the implementation of flexible character recognition equipment involves complex logical functions.

TABLE 1

## INPUT/OUTPUT EQUIPMENT CHARACTERISTICS ANTICIPATED BY 1970

Magnetic tape units	300,000-400,000 char/sec read write rate	2000-3000 char/inch density
Incremental magnetic tape		
Recorders	800-1000 char/sec record rate	800 char/inch density
Readers	500-600 char/sec read rate	556 char/inch density
Punched cards		
Punches	300-500 cards/min punched rate	
Readers	2000-3000 cards/min read rate	
Line printers		
Impact type (multiple copy)	1500-2000 lines/min	64 character types; 132 char/line
Non-impact type (single copy)	3000-5000 lines/min	64 character types; 132 char/line



Research into voice recognition and voice output techniques is being conducted in a number of organizations at this time. Limited voice output equipment capable of outputting canned messages is available now, and some equipment has been demonstrated that is capable of putting together recorded words or, in some cases, syllables to make up a message. However, equipment for truly synthesizing voice output from alphanumeric information and equipment for recognizing spoken messages as computer input are still in fairly early stages of research. It is too early at this time to predict any cost and performance characteristics for devices of this type. However, developments in integrated circuits and batch-fabricated memories will greatly reduce costs for this type of equipment also.

Keyboards have always played an important role as a man-to-machine-language transducer. This is true both of independent input devices (e.g., keypunches) and of input portions of consoles providing man-machine interaction. Present types of electromechanical keyboards have suffered from reliability problems, and have required the user's fingers to operate in a basically flat rectangular area. New types of keyboards are being developed that do not involve mechanically moving parts and that may permit more design freedom from the human factors standpoint; this progress includes development of pneumatic, optic, and piezoelectric techniques.

Solid-state replacements for magnetic tape may improve the speed and reliability available for this type of input/output function, but cost competition with magnetic tapes is questionable. At least two different programs are underway to develop solid-state storage modules that could be plugged into read-write electronic units in a manner somewhat equivalent to placing a reel of tape on a tape unit. If this concept proves feasible and economical, the input/output and off-line storage functions presently provided by magnetic tape could be provided by high-speed, high-reliability devices, and media with no moving parts.

The goals for one development program of this type are 4 million characters per module, read-write rates in the order of 2 or 3 million characters per second, and costs of approximately 0.015¢ per character for off-line storage. A further advantage that would be offered by this particular device is random access (in 1  $\mu$ sec) to any block of data within a storage module on the read-write unit in comparison to the strictly serial access of magnetic tape. The read-write unit would have approximately one-tenth the power requirements and weight of a magnetic tape unit and about half the size. If a device of this type provides random access to a block of data in the storage module, it could also be used as a replacement for electromechanical on-line mass memories such as magnetic discs, magnetic drums, and magnetic card files.

In large data base systems the greatest improvement in the performance of input/output equipment can be achieved by avoiding input/output operations wherever possible. By keeping the data within the system and by capturing data at the source, much of the need for conventional types of input/output equipment can

be reduced. For example, the need for voluminous printed reports can be reduced sharply if the user is operating on line with the processor through an efficient console. When any part of the data base within the system is rapidly available to the user upon request, he will have little need for large reports that are used for occasionally looking up printed results--particularly since these may be out of date by the time they are used. In general, any effort to increase the extent to which systems are "on-line" will tend to reduce the amount of conventional input/output equipment in the system. Achievement of possible improvements in this area will require a combined effort of users, programmers, hardware engineers, and systems planners and designers.

### Logical Components

For the past five to seven years, discrete component semiconductor circuits have dominated the computer data processing field as logic components. A number of alternatives to transistor and diode electronic circuitry have been proposed, but none of these have proven superior for the majority of applications. These alternatives include cryogenic logic, fluid logic, all magnetic logic, and optical logic. Cryogenic and optical logic are yet to be proven feasible. Fluid and magnetic logic offer some advantages in slow speed applications, such as the implementation of control functions in input/output equipment. However, semiconductor components in their new guise of integrated circuit technology will be dominant for the foreseeable future--probably for the next 10 to 15 years at least. The major types of integrated circuits presently under development with characteristics anticipated by 1970 and brief comments on the advantages or disadvantages of each are shown in Table 2. The implications of the circuit speeds shown in Table 2 can be seen by comparing them with the 30 nanosecond propagation delay and 1.6 megacycle clock rate presently quoted for the IBM 360-Model 40, and the 4 nanosecond propagation delay and 5 megacycle clock rate quoted for the large scale Model 70. However, the effects on system cost and reliability, which are not shown in Table 2, will be much greater.

The speeds shown in Table 2 for different types of circuits are chosen to be realistic, but many in the semiconductor industry will consider them overly conservative. Failure rates as low as 0.0005 to 0.0001% per 1000 hours are anticipated. Costs are expected to range between 3¢ and 5¢ per circuit in large interconnected circuit arrays. The cost will be somewhat higher for linear circuits requiring thin-film passive elements and somewhat lower for repetitive functions (e.g., storage arrays) using large MOS arrays. These figures are intended to indicate cost potentials that can be realized by semiconductor technology. However, the ability to achieve these costs is dependent upon the use of large interconnected arrays of circuits and, hence, upon the computer industry's ability to develop logical design and machine organization techniques permitting and utilizing such arrays.

Since packaging and interconnections are major factors in the cost of an integrated circuit, the cost potentials stated above can be achieved only by batch

TABLE 2  
MAJOR TYPES OF INTEGRATED CIRCUITS

<u>Technology</u>	<u>Performance</u> <u>Anticipated by 1970</u>	<u>Comments</u>
Hybrid discrete/ thin-film circuits	1 to 10 ns propagation delay 5 to 20 mc clock rate	Useful where high ratio of passive to active components is required (e.g. linear circuits) and where high power capability is required. Higher cost and probably lower reliability.
Monolithic circuits	0.5 to 10 ns propagation delay 10 to 50 mc clock rate	Low cost, high speed, and high reliability. High value and high tolerance passive components are very difficult, but the use of extra active elements can help compensate for this.
Hybrid monolithic/ thin-film circuits	1 to 10 ns propagation delay 5 to 20 mc clock rate	Compromise between the advantages and disadvantages of discrete components and monolithic circuits. More expensive than monolithic circuits but useful for linear circuits requiring higher tolerance passive components.
Metal-oxide- semiconductor (MOS) circuits	20 to 100 ns propagation delay 2 to 10 mc clock rate	Simpler to make and easier to fabricate large arrays of interconnected circuits. Lower power consumption. Speed approximately one order of magnitude slower than monolithic circuits.
Silicon-on-Sapphire circuits	20 to 100 ns propagation delay 2 to 10 mc clock rate	Fabrication suitable for large arrays. Promising, but presently being actively pursued by only one company.
Active thin-film circuits	Too early to predict	Potentially cheaper and easier to fabricate very large arrays. Feasibility is not proven and utilization much farther away.

fabricating large arrays of interconnected circuits in a single package. This process raises many difficult and conflicting questions, such as packaging design, maintenance philosophy, flexibility, and functional logic segmentation, which time and space do not permit covering here.

### Storage

Several different techniques for batch fabricating solid-state electronic or magnetic storage devices will provide improvements in internal storage costs and reliability compatible with those for integrated circuit components. Very large capacity auxiliary storage requiring electromechanical devices will probably continue to be a problem area.

In considering storage in the broad sense, it is helpful to divide storage requirements into four major categories:

- . Registers and high-speed control memory
- . Main internal memory
- . On-line auxiliary storage
- . Off-line auxiliary storage

Because of the wide difference in characteristics and cost it is also helpful to differentiate between solid-state on-line auxiliary storage and electromechanical on-line auxiliary storage. A particular type of storage technology may be useful in more than one of these categories but the trade-offs between capacity, speed, and cost will vary with the category. The characteristics anticipated for solid-state storage devices in 1970 are shown in Table 3; those for electromechanical auxiliary storage devices are shown in Table 4. The implications of the memory speeds shown in Table 3 can be seen by comparing them with the main internal memory speeds of 2.5 microseconds for the IBM 360-Model 40 and 1 microsecond for the large scale Model 70. However, the effect on system cost and reliability will be much greater. The characteristics shown in Table 4 for electromechanical auxiliary memories compare with present disc file access times of 20 to 200 milliseconds and capacities of 40 to 250 million characters.

Solid-state electronic and magnetic devices are applicable to registers and high-speed control memories, main internal memories, and on-line auxiliary storage, while electromechanical storage devices are applicable primarily to large capacity on-line auxiliary storage and off-line auxiliary storage. As discussed previously, some off-line auxiliary storage devices, such as magnetic tape units, are also considered input/output equipment. In fact, the distinction between off-line auxiliary storage and input/output equipment is somewhat nebulous, based largely upon whether it is used to store information generated by the processor for its later use, to enter data initially into the system from the outside world, or to transfer data from the system to the outside world.

TABLE 3  
STORAGE DEVICE CHARACTERISTICS ANTICIPATED IN 1970

Type of Storage	Registers and High Speed Control Memories		Main High-Speed Internal Memories		Solid-State On-Line Auxiliary Storage Devices		Comments
	Typical Capacity (Words)	R/W Cycle Time	Typical Capacity (Words)	R/W Cycle Time	Typical Capacity (Words)	R/W Cycle Time	
Integrated Cir. Arrays	256	50 ns	0.01x10 <sup>6</sup>	0.2 $\mu$ s			Most promising for very high-speed registers and control memories.
MOS Arrays	512	250 ns	0.02x10 <sup>6</sup>	0.7 $\mu$ s			Promising for low cost intermediate capacities; volatility is disadvantage.
Planar Thin-Film	512	100 ns	0.1x10 <sup>6</sup>	0.5 $\mu$ s	2x10 <sup>6</sup>	1 $\mu$ s	Promising for fast control memories, possibly for on-line aux. storage; questionable for main internal memory
Laminated Ferrite	512	150 ns	0.1x10 <sup>6</sup>	2 $\mu$ s			Reasonable yields not proven for capacities over a few hundred words; actively pursued by only one company.
Plated wire	512	250 ns	0.2x10 <sup>6</sup>	0.5 $\mu$ s	4x10 <sup>6</sup>	1 $\mu$ s	Very promising in all categories.
Magnetic Core Matrix	512	350 ns	0.1x10 <sup>6</sup>	0.7 $\mu$ s	2x10 <sup>6</sup>	3 $\mu$ s	Well established and will be dominant for several years; will be replaced eventually by batch-fab techniques.
Permalloy-Sheet Toroid			0.2x10 <sup>6</sup>	20 $\mu$ s	4x10 <sup>6</sup>	100 $\mu$ s	Potential for very low cost but feasibility and yield are unproven; actively pursued by only one company.
Continuous-Sheet Cryogenic			2.0x10 <sup>6</sup>	2 $\mu$ s	20x10 <sup>6</sup>	5 $\mu$ s	Feasibility still unproven; not economic for capacities below appx. 10 <sup>8</sup> bits because of refrigerant cost.
Ferro-Acoustic			20x10 <sup>6</sup>	(serial)			In early research stages; concept promising for low-cost block-oriented aux. storage, but feasibility not proven.

TABLE 4  
ELECTROMECHANICAL AUXILIARY MEMORY CHARACTERISTICS ANTICIPATED IN 1970

Type of Device	Capacity Per Unit In Char.	Average Access Time	Data Transfer Rate Ch/Sec	On-Line or Off-Line Storage	Comments
Magnetic Drums	$250 \times 10^6$	80 ms	500,000	On-Line	Large physical volume; well proven by field use for years.
Fixed-Head Disc Files	$100 \times 10^6$	20 ms	800,000	On-Line	Fastest access time but highest cost; relatively new with little field experience.
Moving-Head Disc Files	$1,000 \times 10^6$	80 ms	800,000	On-Line	Most field experience of on-line devices; best cost, capacity, and access time compromise.
Removable Disc Files	$50 \times 10^6$	100 ms	500,000	Either	Relatively new but widely accepted; offers advantage of both on-line and off-line capability.
Magnetic Tape Loop	$20 \times 10^6$	80 ms	200,000	Either	New and relatively unproven in the field; made by only one company at present.
Magnetic Tape Reel	$50 \times 10^6$	(serial)	400,000	Off-Line	Well established and proven for many years; lowest cost per character off line; serial access.
Magnetic Card Files	$1,000 \times 10^6$	200 ms	300,000	Either	Available several years, but not as well established as discs; lower cost per char. for large capacity.
Optical Discs	$150 \times 10^9$	Seconds	500,000	Either	New and unproven by field use; offered by only one company; read-only; largest capacity and low cost per character; very slow access.

The costs of storage will vary with speed, capacity, and the particular technique employed. The following costs for given categories of storage, including storage media and all mechanical and electronic components necessary to provide an operating memory, are anticipated by 1970:

Registers and high-speed control memory - 2 to 5¢ per bit.

Main internal memory - 1 to 3¢ per bit.

Solid-state random access on-line auxiliary storage - 0.2 to 1¢ per bit.

Electromechanical on-line auxiliary storage - 0.001 to 0.01¢ per bit.

Photographic on-line auxiliary storage - 0.0005 to 0.005¢ per bit.

The costs shown above compare with present costs of 50¢ to \$10 per bit for registers and high-speed control memories, 5 to 50¢ per bit for main internal memories, and 0.01 to 0.1¢ per bit for electromechanical on-line auxiliary storage.

In addition to their use in storing the data base, storage devices are also used as registers and buffers in peripheral equipment, such as input/output devices and inquiry/display consoles, and as integral parts of the central processor. The main internal memory in the central processor is used for storing requests and active portions of the data base as well as for storing the programs controlling the operation of the system. Hence the concept of storage hierarchies is very important in considering the use and capabilities of storage devices. There is no one ideal type of storage that fulfills all requirements while providing the maximum speed and capacity for the minimum cost.

It is necessary to use a combination of storage devices utilizing the best characteristics of each to effect a better over-all storage system. This is true not only within the processor itself and between the internal and external storage, but also with respect to different levels of external storage. Data base storage will require medium capacity, random access, solid-state, on-line auxiliary storage; large capacity, low cost, on-line auxiliary storage; and very large capacity, very low cost, off-line auxiliary storage. A typical system in the future might combine integrated circuit registers, magnetic core main internal memory, plated wire on-line solid-state auxiliary storage, magnetic disc electromechanical on-line auxiliary storage, larger capacity photographic read-only on-line auxiliary storage, and very large capacity magnetic tape off-line auxiliary storage. One important aspect in the efficient use of hierarchical storage that is needed for development of machine organization and software techniques that make the entire internal and on-line auxiliary storage appear as a single uniform storage to the user.

Although only digital storage is considered here, it is of course possible to provide microfilm or printed documents for bulk files. The computer system can be used to facilitate the location of information in these files by providing indexing and cross referencing.

In the past few years, relatively large development efforts have been expended on associative memories that can address stored information on the basis of a portion of its contents rather than on a unique numeric address. Data is located by association rather than by physical location. Basically, an associative memory involves sufficient logical capability to permit all memory locations to be searched essentially simultaneously (i.e., within some specified memory cycle time). The search may be made on the basis of the entire contents of each location or upon the basis of selected bit positions of each location. Searches may be made on the basis of equality (greater-than-or-equal-to, less-than-or-equal-to), between limits, or, in some cases, more complex criteria.

Associative memories developed to date are significantly more expensive than random access memories having comparable capacity and cycle time. In some types of applications the ability to address the memory by content may offer over-all systems economies or speed improvements that justify the cost of this type of memory. However, most of these advantages can usually be obtained by using a relatively small associative memory in conjunction with a large capacity random access memory. Hence, it is not likely that a central processor will utilize a large associative memory as the main internal memory unless some unforeseen breakthrough in associative memory technology is achieved.

A related concept is that of the associative processor in which logical processing elements as well as storage elements are distributed in associative cells. Since the cost of logical functions in each cell or word must be made very cheap if large associative memories are to be feasible, it is argued that additional logic to provide an associative processor can be added relatively economically. There is some question as to whether a large associative memory can be efficiently utilized unless the processing functions are distributed with the storage cells. It is very unlikely that associative processors will be developed to an adequate state for utilization in data base systems in the foreseeable future. On the other hand, small associative memories used in conjunction with large random access memories may offer advantages in large data base systems that will justify their cost. Such uses might include indexing, cross referencing, and the implementation of complex search criteria for data retrieval.

### Displays

A display screen or visual transducer is an essential part of inquiry consoles for interrogating and utilizing a computer-centered data base system. The keyboard techniques discussed under input/output equipment are directly applicable to the keyboard and manual input portions of the inquiry console, while the logical components and storage techniques discussed previously are applicable for implementing the control and storage functions necessary in the display. A screen size of 15 x 20 inches is probably adequate for such consoles. In many cases, small screen sizes will suffice. Hence, direct view display techniques are completely adequate. Table 5 summarizes the characteristics of display techniques.



TABLE 5  
SUMMARY OF CHARACTERISTICS OF DISPLAY TECHNOLOGIES

<u>Display Technology</u>	<u>Brightness (Ft-Lamberts)</u>	<u>Resolution</u>	<u>Color Capability</u>	<u>Feasibility</u>	<u>Comments</u>
Cathode-Ray Tube	40	Good	Color tube can be used	Readily available	Basic technology for consoles through early 1970s; flexible; not solid state; requires vacuum and high voltages.
Electro-luminescent	20	Limited	Multiple-dot color possible	By 1970, if ever	Requires development of cheap integrated storage inherent to display panel; direct view; matrix addressing.
Opto-Magnetic	Not Available	Good	Color a function of reflection angle	Uncertain	Direct view reflective type display; matrix addressing; promising when feasibility is proven.
Laser-luminescent	Not Available	Good	Unknown	Probably by 1970	Digital positioning promising, but feasibility is uncertain; very attractive when proven feasible.
Injection Electro-luminescence	Not Available	Unknown	Unknown	Uncertain	High brightness expected; matrix addressing; solid state; low voltages; in early research state, but attractive if proven feasible.

The cathode-ray tube represents a well established technology that will probably be dominant through the early 1970s. While the cathode-ray tube is adequate and satisfactory from most standpoints, it has some disadvantages. While these are not critical, they will justify the utilization of other display technologies when these have proven feasible. The major disadvantages of cathode-ray tubes are associated with their incompatibility with new solid-state batch-fabrication technologies. These disadvantages include physical volume, lesser reliability, high power requirements, and a need for high voltage circuits. In some applications involving high ambient light conditions, the brightness and contrast offered by cathode-ray tubes may also be considered limitations. Electroluminescent, opto-magnetic, and injection electroluminescence matrix displays can be implemented in flat panels of considerable less volume. The opto-magnetic and injection electroluminescence matrix displays are particularly attractive because their voltage and power requirements are compatible with those of integrated circuits. Laser luminescent displays do not require a large evacuated envelope but have somewhat the same disadvantage as cathode-ray tubes with respect to physical volume and requirement for higher voltages. This type of display may find application more in large screen displays than in consoles of the type considered here.

#### IMPACT OF HARDWARE TECHNOLOGY

The impact of these new hardware technologies upon computer centered data base systems can be dramatic if the basic component capabilities are properly utilized. On the other hand, the impact will be much less significant if systems designers continue to use the new technologies in the same way that old ones have been used. For example, replacing transistorized circuits directly with integrated circuits will have only an evolutionary impact while utilizing the true capabilities of integrated circuits in the form of large interconnected arrays will provide a revolutionary impact. Some of the effects these new technologies can have and the problems that must be faced and properly solved in their utilization are discussed below.

##### Central Processor

Integrated circuits and batch-fabricated memories will make logic and internal storage very cheap, higher speed, and more reliable. Improvements in the speed and cost of internal storage devices may not be as significant as those for logical components, but internal storage will not limit the performance and capability of future systems. It is anticipated that a portion of the cost and speed improvements in logical components will be utilized to provide greater capability and sophistication in the processor and computing portions of the system. Hence a reasonable balance between central processor logic and internal storage costs will be maintained.

Moderate capacity associative memories at reasonable cost will facilitate indexing, cross referencing, and other storage and retrieval functions. The

combination of low-cost high-speed logic and random access storage with limited amounts of associative memory can provide extremely sophisticated and complex processing, addressing, and editing functions at very low cost compared to the cost of storing the data base itself.

From the standpoint of basic hardware technology, the components and techniques will be available to mechanize very sophisticated logic systems with extremely low cost, small size, and high reliability. Machine organization and system design techniques to take advantage of these component technologies in implementing data base processing requirements must be developed. It will be possible to package the processing and computing capability of an IBM 7094 in a shoe box (without specifying the shoe size) for a few thousand dollars, but the different question is what do we really want to put in that shoe box. With respect to the central processor, the major problems will be determining what functions should be implemented for the user and the proper hardware and software combinations and interaction. If these problems of user requirements and system design are properly solved, processor capability should permit easy manipulation and accessing of a large data base.

#### Inquiry/Display Consoles

For the inquiry/display console, the major problem is again that of determining the proper functions to implement for the user. The basic technologies discussed will permit the implementation of consoles that will greatly facilitate the man-machine interaction. Integrated circuit and storage techniques discussed previously will permit complex control and buffering functions at low costs. The techniques will also permit the incorporation of satellite computing capability in the console. The keyboard and display techniques discussed previously will be quite adequate although the costs will be somewhat high in relation to the logic and storage functions. If display technologies, such as injection electroluminescence matrices, that are compatible with batch-fabrication techniques and low voltage electronics prove feasible, the cost and size should be reduced and the reliability increased.

#### On-Line Auxiliary Storage

Moderate capacity (e.g.,  $10^8$  bits) on-line high-speed random access auxiliary storage will be feasible and economical. This will permit greater utilization of on-line auxiliary storage for large program libraries, index and cross reference files, and data base storage. Very large capacity low-cost electromechanical on-line auxiliary storage will permit fast access to a much larger data base at a reasonable cost, particularly if this storage can be "read-only" so that optical techniques can be used. However, relative to the processor and internal storage, large capacity on-line auxiliary storage will still present a cost problem for large data bases, particularly if the data base must be alterable and on line. The reliability of electromechanical auxiliary storage will not be consistent with that of the solid-state central processor and memory.

### Input/Output and Off/Line Storage

The major problems from the hardware standpoint for future data base systems will be presented by the input/output equipment and off-line storage. These will be critical factors in limiting the capability of future systems. The speed, cost, and reliability of these equipments will be out of line with those of the central processor and on-line auxiliary storage. This will cause a serious imbalance in the over-all system. Unfortunately, very little is being done about this situation relative to the effort being devoted to logical component and memory technologies. This is probably due partially to the fact that the situation is much more difficult and no promising solutions exist, and partially to the fact that input/output had traditionally been a less glamorous field than machine organization and component and memory technology. New types of input/output equipment (e.g., character recognition) offer some promise for the future. However, it will probably be necessary to place major emphasis on system design and organization concepts that minimize the need for input/output equipment.

### PROBLEM AREAS

Several major problems will exist, including:

- . Low cost, large capacity, alterable on-line storage.
- . Low cost, reliable input/output equipment.
- . Proper selection and organization of storage hierarchies.
- . Low cost inquiry/display consoles for man-machine interaction.
- . Economical associative memories or other hardware techniques to facilitate file access by content.
- . Protection of data base in multi-computer, multi-user systems with many remote users.
- . Communications between remote user consoles and the processor, between different processors, and between remote processors and the data base.

The need for solving the real problems must be emphasized. Government and industry must stop glamourizing high speed logic and bigger computers and concentrate on the major problems of the future--input/output equipment and very large capacity, low cost, on-line auxiliary storage.

**COMPUTER-CENTERED DATA BASE SYSTEMS IN SUPPORT OF HIGH MILITARY COMMAND\***

Presented by: Captain Charles W. Turner, CEC, USN  
Office of the Director of Defense Research  
and Engineering

It is a pleasure to be here tonight and to have this opportunity to discuss with you some of the user needs and problems we have in utilizing computer-centered data base systems in support of high military command functions.

I have been asked to comment on how computer-centered data base systems are helping military users, what are the current problems facing the users, how the systems should be improved, and what sorts of administrative and technical solutions are required to get us from today's systems to those of the future.

The military user has learned to accept his vulnerabilities in security, communications and intelligence, while working toward their eventual removal. I do not mean to gloss over these major problem areas. Much has been written about them and the problem of survivability. I could add little to what has been said. Instead, I would like to turn your thoughts to three other areas of interest.

- . Definition of terms
- . The military staff officer
- . The functional program

Better understanding in these three areas is necessary to perform studies of the information processes pervading military staff institutions. Good analytical work is a prerequisite to development of a wider range of automated support for command functions.

Many people feel that this cannot be done. Some even feel that it may not be necessary. My thought is that now is the best time to work on the problem of automated support for command functions. If we were to go to war, development of additional automated support of the military staff would be a priority requirement. If we should return to absolute peace, we would lose manpower in the headquarters, but not the desire for command and active management of affairs from the highest level our communications will reach. Again, increased automated support is a priority need.

---

\* Speech given at symposium dinner.

It takes time to define and solve information problems and establish a data base. We have the time and tools for functional analysis and problem solving. As a consequence, I am convinced that more emphasis must be placed on basic analysis.

If I interpret the purpose of this symposium correctly, it is concerned with the development of improved software tools. One might ask, "For what purpose?" The objective most often stated is "to improve the commander's ability to exercise command and control." We must take another look at this statement and see if it cannot be stated in a more precise way to guide further research.

For the past few years I have been preoccupied with the problems of acquiring improved facilities and data processing capabilities for the unified command authorities who were given strategic direction and operational command responsibilities in the Reorganization Act of 1958. Consequently, I could talk at some length on what the term "command and control" means. As you know, these words mean different things to different people--military and civilian. Each has his own concept and definition, conditioned largely by background, perspective, and current occupation.

The military commander seems to hold this view: Without reducing the explanation to words, he knows that a command and control system is a system for exercising power in support of national policy. He knows that a military command control system, to be effective, must integrate factors of strategy, tactics, intelligence, communications, and logistics into centralized, unambiguous direction of combatant forces.

A military commander is willing to use any tool or technique that will help him exercise command. He will also ignore anything that may, in his estimate of the situation, subtract from his capability to do so.

Command functions do not readily lend themselves to automation. Some features of staff work do. For this reason, mainly, I think that "computational, analytical, and data processing support to command" is more descriptive of what we are doing and can do today with automation and the state of our knowledge of how to apply this new technology in a helpful way in a high command environment. Of course, there are those who do not agree. This tends naturally to stimulate the dialogue with the system designer.

In addition to the problem of defining the term "command and control," the term "system" has a number of meanings. To a military officer a weapon system is a concept within his comprehension. Information-system concepts are not as easily defined. A staff officer has to think conceptually about the military information process in system terms when he attempts to use automation. Most probably, though, he does not think of himself as being in a system.

---

The logician, however, appears to have thought the problem of using this term through. He looks at an argument as a system. To precisely identify the system, his argument, he first defines the universe of discourse. This has a way of placing his argument in the proper perspective.

This is something that we do not do when we talk about command and control systems, or computer-centered data base systems for that matter. Perhaps we should first define the "universe" that goes with the "system" before we attempt to discuss the system. This technique might help us clarify some of the definition problems we face.

To progress, we need better communications between the civilian scientist and the military user. One step that might be helpful in the definition of terms would be to accept the military terms as defined in the Joint Dictionary, JCS Pub. 1, taking action to revise or add new terms when necessary.

How do you do this? The Joint Command and Control Requirements Group in the Joint Staff processes changes in command and control terms. DOD Directive 5000.9 directs use of the Joint Dictionary throughout the Department of Defense in matters of strategy, forces, weapons, and arms control. It was made an unclassified publication to facilitate widespread usage. Why not use it?

It is difficult to talk in an unclassified way on the use of current systems since an assessment of military capability becomes involved. Today we do not know how to do many high command tasks on a computer, and we are limited by the dimensions and content of the data base. In addition, there are other limitations. Therefore, the computer-centered data base system is, at best, a tool to aid the staff in doing its work in support of the commander. The staff reserves the right to use, alter, or reject the outputs, just as the commander has the right to use staff advice and assistance, or ignore it. Contrary to popular misconception, there is no personalized use of computers by high command. I also do not know of any display system that interacts directly with high level command authority. High command just will not push buttons, and this situation may never change.

This intuitive judgment about button pushing has been formulated by the Institute for Defense Analyses into a theorem that goes something like this: 'One thing we do know is that the higher you go in the military hierarchy the more the hardware must be tailored to the individual. The lower you go, the more you can tailor, through proper training and indoctrination, the individual to the machine.'

Is this relevant to our discussion here tonight? The hard fact of life is this: A computer-centered data base system, even with proven software, does not give a commander a command and control system. To assume that it does can lead to serious difficulty for two reasons. First, without the direct

participation of the commander, there cannot be a command and control system. Second, the high command function does not lend itself to the precise definition and repetition necessary for automation to be utilized to full advantage. Further, no existing computer system can be sufficiently flexible to meet the needs of two distinctly different worlds. The needs of the strategic world are quite different from those of the tactical world.

The strategic world looks like an off-line, ad hoc operating environment with few repetitive events. It is largely undefined and unpredictable and relies on fragments of data handled at low data rates. Control is decentralized and operational command is delegated. The strategic world is best studied through induction and intuition, is known only through experience, and is altered by the slow process of evolution.

The tactical world is an on-line, institutionalized operating environment containing some repetitive events. It is better defined, reasonably predictable, and contains formalized data systems operating at high speeds. Control is centralized and command is the personal responsibility of one individual. The tactical world lends itself to direct analysis by deduction and is subject to occasional revolution created by changes in threats and weapons.

Matters might be less complex if you could place a military command clearly in one world or the other, but this is not possible. Today, commands can change worlds on a moment's notice. The change is a function of the situation that must be faced at any given time. Ideally, a computer-centered data base system should be flexible enough to be effectively used in either world, strategic or tactical.

How should these systems be improved? Concentrate on flexibility and reliability. Specifically, design the man-machine interface to the data base so that the staff officer who is a subject matter specialist can deal directly with the data base rather than through an interpreter. Provide him with a query language he can learn and use. Develop an information retrieval capability that has some selectivity and is at least as easy to search as a file folder.

Thus far we have put considerable emphasis on the development of general-purpose data management systems to provide better tools to work on problems in the functional software. Solutions will not come easily. The information processes of command are more akin to the social sciences than to the natural sciences. Staff procedures do not always lend themselves to analysis in mathematical terms. Statistical probability methods are being applied with some success as an aid to human judgment.

However, to develop solutions, at some point in time emphasis must shift to the analysis of staff work. To do this type of analysis the scientist must be able to work effectively with the staff officer dealing with the problem on a daily basis. How do we build up a number of small specialized teams to solve problems



that can be used as stepping stones or building blocks to more elaborate and sophisticated structures? How do we avoid creation of intricate management structures which propose to (and occasionally are allowed to) undertake the development of complex solutions to partially understood problems? These are key questions.

The success one can achieve in developing a command support capability which will operate effectively is directly proportional to the extent to which we are able to get the commander and his senior staff officers directly involved in development process. The logical corollary to this statement is that the primary limitation on the rate of acquisition of a functional capability in those areas amenable to automation is determined by the number, time, and capability of military staff officers, with primary responsibilities for operations, intelligence, logistics and communications functions, who can be diverted to assisting in the system development.

Without support and assistance of the commander and his senior staff officers, the system analysts and computer programmers will never discover what is real and what is fictitious about the organization they are trying to develop automated support for. They will solve problems which never existed or which cease to exist before a computer solution for them can be implemented. The point I want to make is that the decision to get involved with a computer is the decision to get the commander involved. He will get in there sooner or later. Happier, if sooner. Unhappier, if later.

We know that the limiting factor is not hardware--it is our ability to design software. We know that we must not experiment with new hardware while we are experimenting with new software. We know we cannot generalize about the "optimum" approach to meeting high command-decision data requirements, yet we seem to.

The limiting factor on progress is experience--experience on the part of the scientists in translating their ideas into procedures for producing data that can be used in reports, studies and plans. Most important, the staff must have experience in the use of computer products in the performance of their daily tasks. To produce this experience, tasks should be programmed at the level of sophistication natural to the operation and the staff knowledge of the operation. By keeping it simple initially, the staff will advance sooner to the stage where their demands on the hardware and software are more than local talent is able to satisfy. At that time you can be certain that the staff will be well aware of the pitfalls and problems, have developed a healthy attitude toward solving them, and be able to give direction and guidance to a technical support activity assigned to assist in the development of an improved capability.

The limiting resource in attaining this experience is the number of staff officers who will devote enough time and attention to planning computer

applications and to trying them out. No amount of money, priority, technology, or level of technical support effort will attain any real progress if this ingredient is missing. Further, it is not possible to accelerate progress beyond the capabilities of the staff officer to contribute to the programming effort.

The intellectual problems in the functional area are formidable. There are no simple solutions. The process of developing solutions and acceptable alternatives includes a number of human relations hazards. Military staff officers and civilian scientists usually have only one thing in common when they first meet to discuss a problem. They are human beings. The differences start at this point.

The staff officer does his work without public recognition and in the name of the commander. He is capable, dedicated, and proud. He welcomes advice, assistance, and new ideas. Exercises, tests, and evaluations bring criticism of the existing staff system and procedures. These are accepted so long as the information is held within the staff and the necessary adjustments can be made as a part of the administrative routine.

Doing analytical and experimental work in a military staff is not easy for a civilian scientist who is accustomed to the freedom and informality of the laboratory. To be able to work effectively on military information problems and to be accepted by the staff is no small accomplishment. The association with the military can also be a rewarding experience.

There are three situations that, whenever they occur, tend to destroy confidence and make further analysis difficult, if not impossible. The situations arise from one or a combination of the following:

- Open publication of the results of the research without clearing with the commander.
- Gratuitous evaluations of military plans and capabilities outside the staff group you are working with.
- Participation in informal information exchanges outside the staff group you are working with without first consulting the senior officer of the group.

If these situations are avoided and you have wisdom, patience, understanding, and a good sense of humor, in addition to your high scientific and technical qualifications, you will find many opportunities to work on the solution of some very challenging problems, whether or not the final result is an additional application for a computer.

1 December 1965

2-35

TM-2624/100/00

(Page 2-36 blank)

Bismarck used a two-by-two matrix for classifying military officers. The parameters were intelligent, stupid, industrious, and lazy. He found that the intelligent and lazy officer could achieve the highest command, that the intelligent and industrious made good staff officers, and that he could always find a use for the stupid and lazy ones. The "industrious and stupid," however, he had to get rid of because they caused trouble.

If this criterion is applied to computers in the high command environment, the device alone no doubt falls into the category of the "industrious and stupid."

It will take a good operations research and programming team working with the device for some time to raise the classification to "intelligent and industrious." Today, uses for the device can be found. In the future, the device may have the capability of replacing a staff officer. Obviously, the commander need have no fear.

SECTION IIIFIVE APPROACHES TO THE SAME DATA BASE PROBLEM

In order to demonstrate the capabilities and advances of current data base systems, four representative systems--essentially dissimilar in structure and purpose--were invited to demonstrate how they would handle the identical data base problem, furnished in advance. In these proceedings, seven systems are described. General Electric's Integrated Data Store volunteered to work the problem, and the COLINGO paper turned out to cover three systems instead of one. The data base problem, as furnished to the participants, is also included beginning on page 3-5.

This session was not intended to be competitive, and, indeed, the approaches are sufficiently varied and unique that an evaluative comparison would be difficult. The seven papers do offer considerable insight into the power and limitations of various approaches and should be instructive to the designer of data base systems.

Introduction.....	3-3
Description of the Data Base Problem.....	3-5
Three COLINGO-like Approaches to the Data Base Problem.....	3-31
COLINGO D.....	3-32
COLINGO C-10.....	3-59
ADAM.....	3-87
Mark III File Management System.....	3-123
On-Line Data Management System for the Massachusetts General Hospital.....	3-143
BEST System.....	3-185
Integrated Data Store.....	3-231

## FIVE APPROACHES TO THE SAME DATA BASE PROBLEM

Introduction: Richard G. Canning  
Canning Publications, Inc.

The goal of this session is to obtain added insight into the structure of generalized file-processing software systems. We hope to obtain this insight through hearing four different approaches to the same data base problem, and through the chance to question the four participants on the choices they have made. In addition, a fifth approach to this problem was submitted by General Electric of Phoenix, Arizona. A report of their Integrated Data Store approach is also included.

At the outset, I would like to emphasize that this session is in no sense a competition. The systems that will be discussed were selected because they are different, and because they represent differing philosophies of approach. We recognize that the systems were designed with different end objectives in mind. The subject area is new and is one in which we all have much to learn. We trust that the questions that follow each talk will be phrased in this spirit.

The participating companies, and the individuals that are representing them, deserve the thanks of all of us. This is an unusual session, and we appreciate their willingness to participate in it.

Other organizations have indicated their interest in participating and have offered to solve the problem on their systems. Time constraints on this session did not allow for more than four presentations. The problem statement was supplied to such organizations, however, along with the invitation to submit their results for incorporation in the proceedings of the symposium. As I mentioned earlier, General Electric has also run the problem.

We have tried to develop a simple, yet challenging, problem that will exercise certain common data base functions--the functions of file creation, file maintenance, subsetting the data base for different types of reports, sorting, simple computation, and the production of printed reports with varying formats. These functions are common to many military and commercial applications, and together represent a significant portion of the programming and system design time for implementing applications. We tried to make the problem simple, so that its solution would require relatively few man-hours on the part of the participants. The problem was developed by a team consisting of Chris Shaw, Al Vorhaus, Bill Crossley, and Claude Baum, of System Development Corporation, and myself.

Since each of you has received a preprint of the problem statement, I will review only the highlights of the problem. The problem is based on a hypothetical organization--two departments in a division of a large company. Two general types of interrelated data records are involved: (a) personnel records for approximately 100 people, and (b) organization unit records for 29 units.

To make the problem representative of real-life data bases, we included four important characteristics of data records:

- Variable-length fields (name fields)
- Repeated fields (skill fields)
- Repeated groups of fields (authorized complements of organization units)
- Hierarchy relationships, between organization units.

We treated the problem as if it involved two main files--a personnel file and organization unit file--plus a table of job and skill code names. However, the participants were not limited to this approach; they could treat the data base as one file, or more than two files, if such suited their system better.

In developing their solutions, however, we asked that the participants treat the problem as if the data base were much larger--many more records and larger size records. Similarly, while many types of transactions could affect these records in an actual situation, we postulated only three typical updating transactions.

The participants were informed from the outset that they were not committed to working the whole problem, if such proved inconvenient for their systems. They were told to concentrate on what their systems could handle, and either skip the rest or supply their own coding. If some of the data supplied proved to be unwieldy, they were to specify the restrictions that would make it more acceptable to their systems and then operate with the modified data. If the suggested report formats did not fit their systems well, then they were to use formats that were more suitable.

### DESCRIPTION OF THE DATA BASE PROBLEM

(As furnished to the four participating systems)

This problem is designed to exercise certain basic functions of a data management system. These functions include file creation, file maintenance, subsetting the data base for producing different types of reports, sorting, simple computation, and the production of printed reports with varying formats.

The data base to be used is small. It consists of approximately 100 personnel records, with about ten types of fields per record, plus 29 organization records, each of which has ten types of fields.

We have tried to make the problem simple yet challenging, so that its solution will require relatively few man-hours on the part of the participants.

#### Data Characteristics

The records in the data base include the following characteristics:

- a) Variable-length fields (name fields).
- b) Repeated fields (skill fields).
- c) Repeated groups of fields (authorized complements of organization units).
- d) Hierarchical relationships, between organization records.

We have prepared the problem as if there were two files--a Personnel file and an Organization file. However, you need not follow this approach; you can use more files or just one file, if such would fit your system better.

While the problem data base is small, the problem should be approached as if the data base were much larger. When organizing the file(s), assume that there are thousands of Personnel records and hundreds of Organization records. Further, assume that each type of record can have over one hundred different fields and a maximum record length of several hundred characters. However, for the problem, it is not necessary to create records of this size; just use the fields that are given. In an actual data base, there would be more instances of variable fields, repeated fields, and repeated groups of fields. Blank fields also would exist in many records.

Also, assume that many types of transactions can affect the Personnel and Organization records. Such transactions would include personnel hired, terminated, transferred; change of skills, salary, job; organization units created, merged, terminated; budget changes; and so on. In the problem, we have included only a few typical transactions.

#### The Data Base

The data base consists of:

- a) The data base listing, Attachment 11.
- b) The table of job/skill codes and names, Attachment 4.

We can provide you with the data base in punched card form, according to your specifications, if you desire. Just tell us how you want the records punched.

You may carry the organization unit names, skill names, and position titles in the records, or in tables (and use table lookup), or in separate files, as you desire.

### Sequence of Steps

Please approach the problem in the following sequence:

- Step 1: Create the file(s).
- Step 2: Produce the Control Report.
- Step 3: Update the file(s) with the three transactions.
- Step 4: Produce the two Periodic Reports.
- Step 5: Produce the two Demand Reports.
- Step 6: Produce the Exception Report.
- Step 7: Produce the Hypothesis Testing Report.

If your system will not handle the entire problem, concentrate on what can be done and skip the rest, or supply your own coding. If some of the data prove unwieldy, specify the restrictions that will make the data acceptable to your system and operate with the modified data. If the suggested formats do not fit your system well, use formats that are more suitable.

### Information Requested About Your Solution

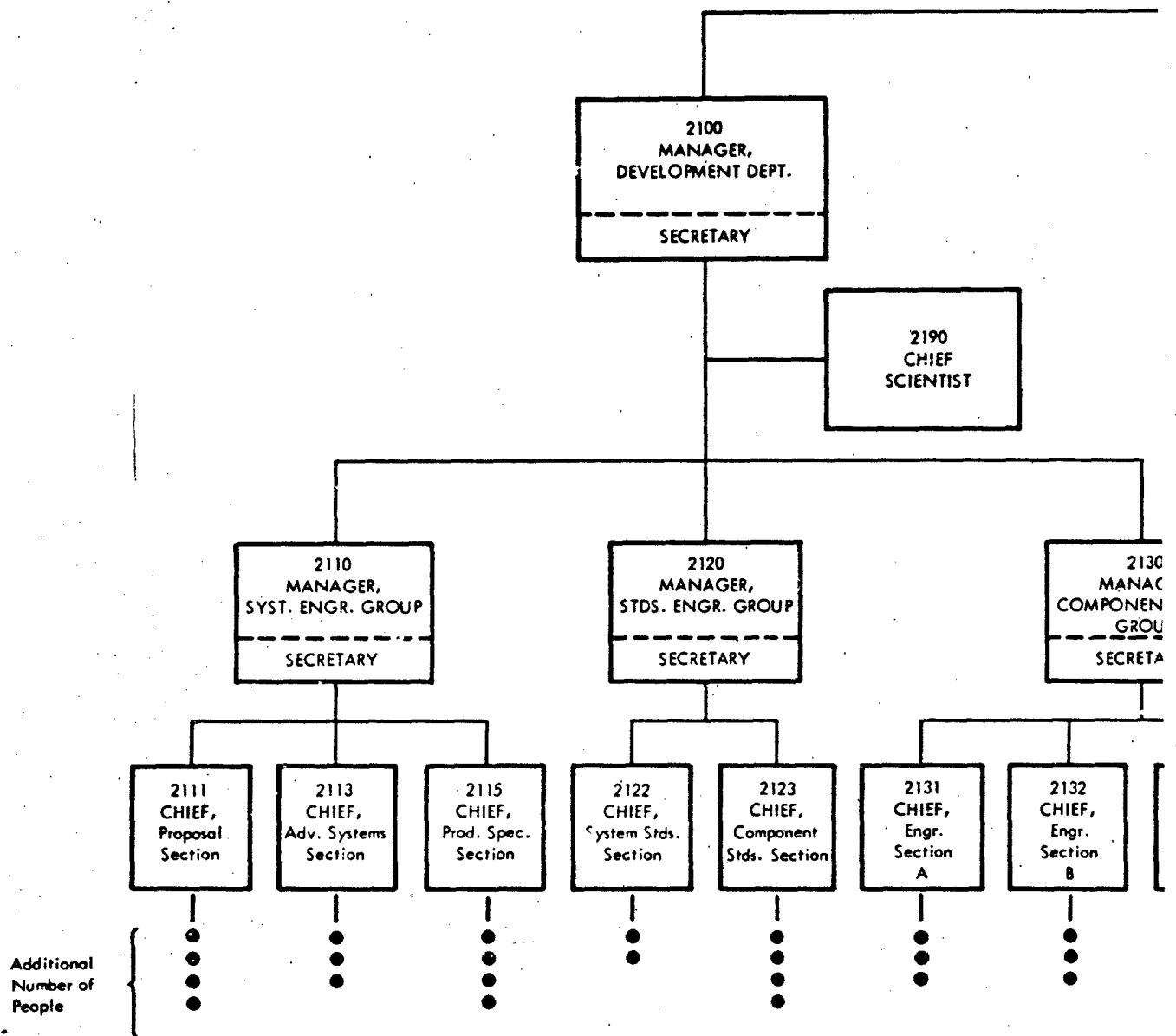
You have your choice of running the problem on the computer, doing it manually, or merely indicating how your system would have handled it. Whichever approach you choose, we would like the following information presented at the symposium and also provided for incorporation in the Proceedings of the symposium.

1. Discuss the file organization you have chosen for the two types of records, and the reasons for your choice. Discuss your handling of variable-length fields, repeated fields, repeated groups of fields, and hierarchical relationships.
2. Show the code sheets you used for file creation and file maintenance.
3. Show the retrieval requests expressed in your system's query language.
4. Give a list of the runs required for producing each of the reports. Indicate whether your system can batch the two demand reports and can perform the selection for the two reports on one pass of the file.
5. If your report formats deviate from the suggested formats, show the formats used.

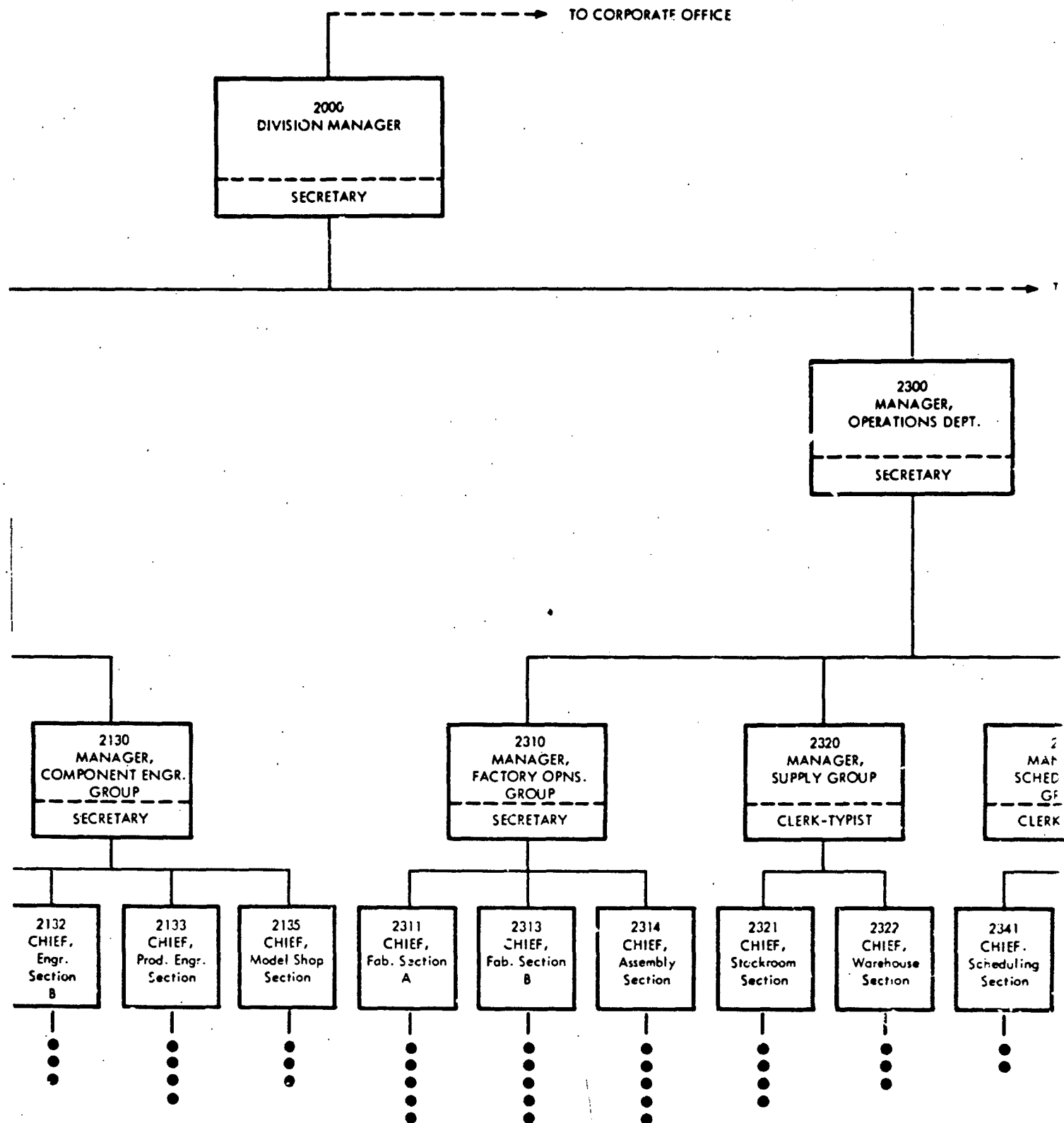


List of Attachments

<u>Attachment</u>	<u>Name</u>
1	Organization chart, on which the problem is based.
2	List of Organization Units.
3	Glossary; definition of the fields in the Personnel and Organization records.
4	List of Job/Skill codes and names.
5	Control Report.
6	Updating transactions.
7	Periodic Reports.
8	Demand Reports.
9	Exception Report.
10	Hypothesis Testing Report.
11	Data Base listing.



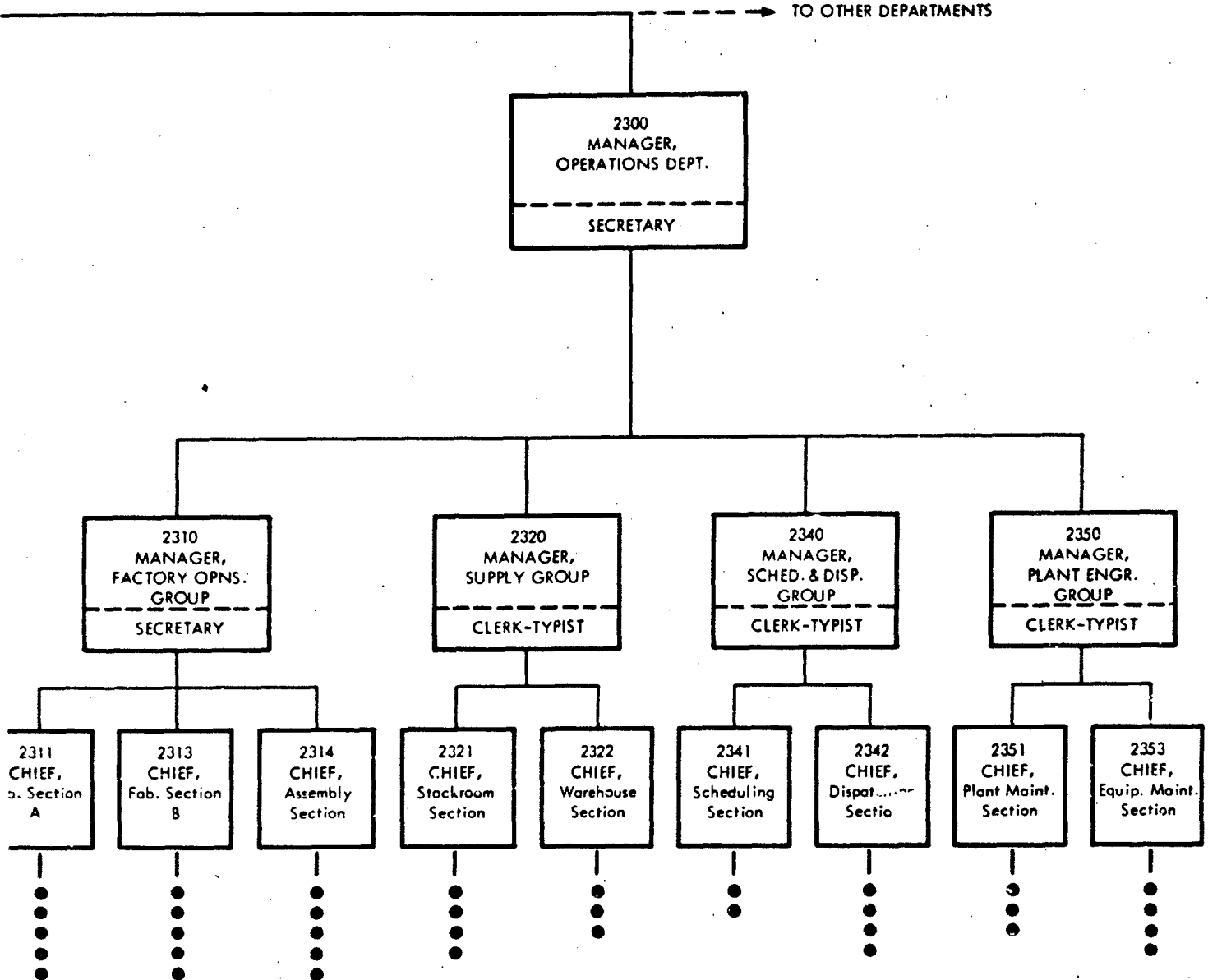
ATTACHMENT 1  
ORGANIZATION CHART  
REPRESENTATIVE DIVISION



TO CORPORATE OFFICE

GER

TO OTHER DEPARTMENTS



C

Attachment 2List of Organization Units

<u>Code</u>	<u>Unit name</u>
1000	Corporate Office
2000	Representative Div.
2100	Development Dept.
2110	Syst. Engr. Group
2111	Proposal Section
2113	Adv. Systems Section
2115	Prod. Spec. Section
2120	Stds. Engr. Group
2122	System Stds. Section
2123	Component Stds. Sec.
2130	Component Engr. Group
2131	Engr. Section A
2132	Engr. Section B
2133	Prod. Engr. Section
2135	Model Shop Section
2190	Chief Scientist
2300	Operations Dept.
2310	Factory Opns. Group
2311	Fab. Section A
2313	Fab. Section B
2314	Assembly Section
2320	Supply Group
2321	Stockroom Section
2322	Warehouse Section
2340	Sched. & Disp. Group
2341	Scheduling Section
2342	Dispatching Section
2350	Plant Engr. Group
2351	Plant Maint. Section
2353	Equip. Maint. Section

Attachment 3GlossaryPersonnel Information (Example is provided at end of this Attachment)

**Name:** Name of the person; 10 to 25 alphabetic characters, consisting of 1, 2, or 3 initials plus last name; you can use a special symbol to separate initials from the last name, if you wish.

**Number:** A 5-digit employee number; this number has no reference to job level or alphabetical listing of the names.

**Unit:** A 4-digit code representing the organization unit to which the person is assigned; see Organization Chart, Attachment 1.

**Job code:** A 4-digit code designating the skill specified for the position that this person occupies; it is not mandatory that a person's job code match one of his skill codes. See the list of job/skill codes and names, Attachment 4.

**Level:** A 4-alphabetic character code: "HEAD" for head of this particular organization unit, "EMPL" meaning employee, and "CONS" meaning consultant. (This alphabetic code is included to exercise selection using alphabetic characters.)

**Position Title:** Title of the position held. For "EMPL" level, title agrees with the job code name. For "CONS" level, title is "Consultant." For "HEAD" level, title is "Chief" for Sections, and "Manager" for Groups, Departments, and Divisions.

**Salary:** Person's current annual salary.

**Skill code:** A repeated field of 4-digit code(s) designating the person's skills. See list of job/skill codes and names, Attachment 4. A person must have a single primary skill, which usually agrees with the person's job code. In addition, a person may have from 1 to 9 secondary skill codes.

**Skill name:** A 20-alphabetic character field giving the skill name associated with each skill code; see Attachment 4.

**Sex:** The person's sex, coded M or F.

**Birthdate:** The birthdate is a 6-digit field, in the sequence month-day-year. For some retrievals, it may be necessary to select on any one or two of these three subfields.

Attachment 3 (continued)Organization Information (Example is provided at end of this Attachment)

Org. Unit: A 4-digit code representing the organization unit; see the Organization Chart, Attachment 1, and the list of organization units, Attachment 2.

Reports to: A 4-digit code representing the organization unit to which this unit reports.

Org. Name: A 20-alphabetic character field giving the name of this organization unit.

Authorized complement--the heading for the following repeated groups of fields:

Job code: A 4-digit code designating a position assigned to this organization unit; see list of job/skill codes, Attachment 4.

Unit code: A 4-digit code designating a sub-unit assigned to this organization unit; see list of organization units, Attachment 2.

(Note: One of these groups of fields can have a job code entry or a unit code entry, but not both; see the example below to see the logic of this.)

Title: A 20-alphabetic character field describing the position title (for a job code) or the organization name (for a unit code).

Quantity: A 2-digit field designating the quantity of this job code or unit code authorized for this organization unit.

Salary: A 6-digit quantity field which contains the total authorized annual salaries for this job code or this unit code, in this organization unit.

Total budget: A 7-digit field which contains the total of all authorized salaries for this organization unit.

Attachment 3 (continued)Example of Personnel record

Name: Peterson, N. M.  
Number: 45584  
Unit: 2000  
  
Job code: 0110  
Level: HEAD  
Position Title: Division Manager  
  
Salary: 28,000  
Sex: M  
Birthdate: 06/07/18  
  
Primary skill code: 0110  
Secondary skill codes: 6130, 6625, 6040

Example of Organization record

Org. Unit: 2110  
Reports to: 2100  
Org. Name: Syst. Engr. Group

## Authorized complement:

1110	Manager	1	17,500
5210	Secretary	1	4,200
2111	Section	1	52,600
2113	Section	1	49,000
2115	Section	1	42,000

Total budget: 165,300



1 December 1965

3-15

TM-2624/100/00

Attachment 4

List of Job/Skill Codes and Names

<u>Code</u>	<u>Name</u>	<u>Code</u>	<u>Name</u>
0110	Admin, Division	3510	Assembler
0130	Admin, Department	3540	Dispatcher
0150	Admin, Group	3550	Expediter
		3570	Dispatch Asst
1110	Systems Engr		
1120	Mech Engr	3740	Chief Stock Clerk
1124	Structures Engr	3745	Stock Clerk
1127	Power Engr	3780	Inventory Clerk
		3790	Warehouseman
1130	Elec Engr		
1135	Commun Engr	5210	Secretary
1150	Prod Engr	5220	Clerk-typist
		5520	File clerk
1175	Metallurgist		
1180	Chemist	6040	Reg Sales Mgr
1190	Scientist	6130	Branch Sales Mgr
		6625	Salesman
1330	Draftsman		
1350	Cost Estimator	7110	Maint Engr (Pl)
1351	Mech Techn	7115	Air Cond Engr
1355	Elec Techn	7120	Maint Engr (E)
1365	Tool Designer	7145	Plating & Paint Engr
3125	Tool Maker	7310	Maint Tech (Pl)
3320	Scheduler	7320	Maint Tech (E)
3340	Machine Opr	7350	Air Cond Tech
		7360	Plating & Paint Tech
3345	Milling Mach Opr		
3355	Plating Opr		
3360	Heat Treat Opr		
3370	Painting Opr		

1 December 1965

3-16

TM-2624/100/00

Attachment 5

Control Report

1. Request:

After the file has been created, develop control totals of the number of persons employed, the total of the actual annual salaries, and the number of organization units.

2. Format:

NUMBER OF PERSONS EMPLOYED \_\_\_\_\_

TOTAL ACTUAL ANNUAL SALARIES \_\_\_\_\_

NUMBER OF ORGANIZATION UNITS \_\_\_\_\_

1 December 1965

3-17

TM-2624/10G/00

Attachment 6

Updating Transactions

1. 33144 Quinn, S. M. Transaction: Terminated.
2. 91152 Garber, B. E. Transaction: Change salary to \$8500.
3. 85657 Lee, R. E. Transaction: Add secondary skill code 3570.

Attachment 7Periodic Reports

## 1. Retrieval requests:

Combine the Personnel information with the Organization information, so as to prepare two reports. One report covers all Sections, in ascending organization unit number sequence, showing for each Section its Authorized Complement, Actual Complement, and Deviation from Budget. The second report presents this same type of information for all Groups.

## 2. Format for Section Report:

ORG UNIT        2115  
REPORTS TO     2110

ORG NAME        PROD. SPEC. SECTION

## AUTHORIZED COMPLEMENT

1110	CHIEF	1	12,000
1120	MECH ENGR	1	11,000
1130	ELEC ENGR	1	11,000
1330	DRAFTSMAN	1	8,000
TOTAL		4	42,000

## ACTUAL COMPLEMENT

1110	CHIEF	1	12,000
1120	MECH ENGR	1	11,000
1130	ELEC ENGR	2	21,500
1330	DRAFTSMAN	1	8,000
TOTAL		5	52,500

DEVIATION FROM BUDGET		10,500
-----------------------	--	--------

TM-26 24/100/00

DEVIATION FROM BUDGET	9,500
-----------------------	-------

Attachment 8Demand Reports

## 1. Retrieval requests:

Select records for two reports, as specified below; print the two reports separately. Sort and print the reports alphabetically, according to the format shown below.

Report: Select all records meeting the following criteria:

Any skill code = 3340  
and total number of skills > 2  
and sex = M  
and level ≠ HEAD  
and birthdate > 1916  
and birthdate < 1935

From selected records, extract name, employee number, organization unit, and skill codes; extract skill names if in record, or perform lookup of skill names at some convenient point.

Report: Select all records meeting the following criteria:

Sex = M  
and level ≠ HEAD  
and [ any skill code = 1110  
or any skill code > 1129 and < 1140 ]

From selected records, extract name, employee number, organization unit, and skill codes; extract skill names if in record, or perform lookup of skill names at some convenient point.

## 2. Format:

NAME	NUMBER	UNIT	SKILLS
BOYD, W. V.	15052	2135	1351 MECH TECHN 3340 MACHINE OPR 7320 MAINT TECH (E) 3370 PAINTING OPR
COATES, C. L.	81130	2313	3340 MACHINE OPR 3345 MILLING MACH OPR 3360 HEAT TREAT OPR
FLETCHER, M. W.	21475	2133	1365 TOOL DESIGNER 3125 TOOL MAKER 3340 MACHINE OPR 7320 MAINT TECH (E)

1 December 1965

3-21  
(Page 3-22 blank)

TM-2624/100/00

Attachment 9

Exception Report

1. Retrieval request:

Combine the Personnel information with the Organization information so as to prepare an Exception Report. This report pertains to Sections only, and does not cover Groups or Departments. Select all Sections where the actual number of persons is equal to or greater than the authorized number of persons, and where the actual total of salaries deviates from the budgeted salaries by  $\pm$  \$800 or more.

2. Format:

The format and information presented in this report would be the same as shown on the example of the Section Report, Attachment 7. In fact, the Section shown in that example should occur on the Exception Report.

Attachment 10Hypothesis Testing Report

## 1. Retrieval request:

Considering only Department 2100, transfer all draftsmen positions (job code 1330) in the Department to Section 2123. Note that this involves only the Organization records and not the Personnel records. Note too that this is a proposed change, so that the original Organization records are not to be destroyed.

Prepare revised Organization records for the proposed new organization, covering all Sections and Groups within the Department. Develop new budget figures for this proposed organization.

Prepare a report that shows for every organization unit within the Department, and in the sequence defined below, the present organization and budget, and the proposed organization and budget.

## 2. Format:

The format for Sections is to be the same as shown on the Section Report, Attachment 7, with two differences:

- a) The term "Actual Complement" is to be replaced by "Proposed Complement."
- b) The line "Deviation from Budget" is to be omitted.

The format for Groups is to be the same as shown in the Group Report, Attachment 7, but with the same two differences just noted for Sections.

Print the report in the following organization unit number sequence:

Section	2111
Section	2113
Section	2115
Group	2110
Section	2122
Section	2123
Group	2120
Section	2131
Section	2132
Section	2133
Section	2135
Group	2130
Section	2190
Department	2100



## Attachment 11. Data Base: Personnel (Sheet 1).

EMPL. NO.	NAME	UNIT	JOB CODE	LEVEL	POSITION TITL	SEX	BIRTH-DATE	PRIMARY SKILL	SECONDARY SKILLS	SALARY
45584	PETERSON, N. W.	2000	0110	HEAD	DIVISION MANAGER	M	060718	0110	6130 6625 6040	28000
32579	LYAN, W. R.	2000	0210	EMPL	SECRETARY	F	012143	5210	5420	6000
57060	CARR, P. I.	2100	1110	HEAD	MANAGER, DEVELOPMENT DEPT.	M	072025	1110	1130 1135 0130	24000
15374	CALLAGHAN, R. F.	2100	0210	EMPL	SECRETARY	F	060645	5210	5420	5400
10261	GUTHMAN, G. J.	2110	1110	HEAD	MANAGER, SYST. ENGR. GROUP	M	111020	1110	1130 1135	17500
72556	WARETS, D. L.	2110	0210	EMPL	SECRETARY	F	051745	5210	5420	4200
24188	WALTERS, R. J.	2111	1110	HEAD	CHIEF, PROPOSAL SECTION	M	020216	1110	1120	14000
21675	SCARBOROUGH, J. B.	2111	1120	EMPL	MECH ENGR	M	091414	1120		10500
18130	PENDERSON, R. G.	2111	1130	EMPL	ELEC ENGR	M	012124	1130		11500
91152	GARBER, R. E.	2111	1330	EMPL	DRAFTSMAN	M	070734	1330		8200
30793	CUMPTON, D. R.	2111	1350	EMPL	COST ESTIMATOR	M	028819	1350	1351 1355	8100
81590	FRIEDMAN, J. W.	2113	1110	HEAD	CHIEF, ADV. SYSTEMS SECTION	M	031726	1110	1130	13000
21777	FRANCIS, G. C.	2113	1110	EMPL	SYSTEMS ENGR	M	111122	1110	1130	12000
13581	FALKNER, W. W.	2113	1120	EMPL	MECH ENGR	M	062130	1120		12000
13581	FALKNER, W. W.	2113	1130	EMPL	ELEC ENGR	M	121633	1130		11000
22959	BRIGGS, G. R.	2115	1120	EMPL	MECH ENGR	M	010920	1120		11000
29414	ARTHUR, P. J.	2115	1130	EMPL	ELEC ENGR	M	071940	1130		10500
92802	AGAN, A. J.	2115	1130	EMPL	ELEC ENGR	M	072935	1130		11000
37113	ARNETTE, L. J.	2115	1330	EMPL	DRAFTSMAN	F	101039	1330		8000
63633	BLANK, L. F.	2115	1330	EMPL	DRAFTSMAN	F	101039	1330		8000
48840	GODDARD, D. W.	2120	1120	HEAD	MANAGER, STDS. ENGR. GROUP	M	011923	1120	1135	14500
71160	GPRISM, C. S.	2120	0210	EMPL	SECRETARY	F	033146	5210		4000
35401	ANDERSON, R. E.	2122	1120	HEAD	CHIEF, SYSTEM STDS. SECTION	M	022828	1120	1135	11500
91589	TAUBER, J. S.	2122	1120	EMPL	MECH ENGR	M	111435	1120		9500
80823	STADERMAN, P. K.	2122	1130	EMPL	ELEC ENGR	M	020733	1130		9700
46937	SMITH, R. E.	2123	1120	HEAD	CHIEF, COMPONENT STDS. SECTION	M	117930	1120	1127	11000
82166	RODES, J. H.	2123	1120	EMPL	MECH ENGR	M	030330	1120		9500
28654	LAYMOND, W. F.	2123	1130	EMPL	ELEC ENGR	M	121624	1130		10000
17848	HUGHES, J. W.	2123	1330	EMPL	DRAFTSMAN	F	012025	1330		8500
33144	QUINN, S. M.	2123	5520	EMPL	FILE CLERK	F	111846	5520		5500
48464	CANDLER, W. R.	2130	1120	HEAD	MANAGER, COMPONENT ENGR. GROUP	M	101925	1120	1135	16500
44432	PERGMANN, R. I.	2130	0210	EMPL	SECRETARY	F	071044	5210		4500
68795	COOPER, J. C.	2131	1120	HEAD	CHIEF, ENGR. SECTION A	M	030919	1120	1124	12000
30464	RUSTEN, A. R.	2131	1120	EMPL	MECH ENGR	M	071928	1120		11000
79935	BOLD, E. W.	2131	1120	EMPL	MECH ENGR	M	041636	1120		10500
62999	NORMAN, W. K.	2131	1330	EMPL	DRAFTSMAN	M	110123	1330		9000
90816	MCCARTHY, J. K.	2132	1130	HEAD	CHIEF, ENGR. SECTION B	M	102033	1130	1135	11500
36472	MASSEARD, L. R.	2132	1130	EMPL	ELEC ENGR	M	101010	1130		10000
10295	STILCOTT, D. N.	2132	1130	EMPL	ELEC ENGR	M	022333	1130		9500
89876	STINER, P. P.	2132	1130	EMPL	ELEC ENGR	M	041938	1130		10000
47289	SCRAWNER, F. G.	2133	1150	HEAD	CHIEF, PROD. ENGR SECTION	M	090932	1150	1125 1365	10500
12245	VENT, H. J.	2133	1150	EMPL	PROD ENGR	M	012516	1150	1340	9800
50498	LEGGETT, P. F.	2133	1150	EMPL	PROD ENGR	M	051723	1150	1365	10000
57084	TYNERMAN, A. R.	2133	1330	EMPL	DRAFTSMAN	M	062240	1330		8500
21475	FLITCHER, M. J.	2133	1365	EMPL	TOOL DESIGNER	M	072921	1365	1325 1365 3370	9500
26934	HAWLEY, C. P.	2135	1125	HEAD	CHIEF, MODEL SHOP SECTION	M	061310	1125	1340 1375 1365	9000
15052	ROYD, W. V.	2135	1351	EMPL	MECH TECHN	M	091732	1351	1340 7320 3370	9000
21261	CURTIS, K. R.	2135	1355	EMPL	ELEC TECHN	M	030426	1355	1340 7320	8500
74000	CHAY, S. O.	2135	1325	EMPL	TOOL MAPER	M	070313	1325	1340 7320	9500
87354	DUFFRICH, L. J.	2190	1190	HEAD	CHIEF SCIENTIST	M	042315	1190	1130 1120 1110	22000
87277	SPENFELDT, W. R.	2300	0130	HEAD	MANAGER, OPERATIONS DEPT.	M	050113	0130	1340 7120	20000
61706	PERSCOTT, H. C.	2300	0210	EMPL	SECRETARY	F	070344	5210	5420	5000
61284	MATTHEW, J. R.	2310	0150	HEAD	MANAGER, FACTORY OPNS. GROUP	M	112621	0150	1340 13740	14000
73683	MELCHERT, F. F.	2310	0210	EMPL	SECRETARY	F	081325	5210	5420	5000
19357	PAYNE, W. F.	2311	0340	HEAD	CHIEF, FAB. SECTION A	M	082123	3340	1345	10000
32924	LEVITT, P. S.	2311	3340	EMPL	MACHINE OPR	M	042234	3340	3345	8000
42055	LITTLE, E. F.	2311	3340	EMPL	MACHINE OPR	M	030228	3340	3360	8500

1 December 1965

3-25

TM-2624/100/00

## Attachment 11. Data Base: Personnel (Sheet 2).

EMPL. NO.	NAME	UNIT	JOB CODE	LEVEL	POSITION TITLE	SEX	BIRTH-DATE	PRIMARY SKILL	SECONDARY SKILLS	SALARY
52467	MILLER, F. L.	2311	3340	EMPL	MACHINE OPR	M	080820	3350	3510 7320 3970	9000
28347	KOPELY, S. E.	2311	3340	EMPL	MACHINE OPR	M	082036	3340	3155	6500
85657	KFF, R. F.	2311	3340	EMPL	MACHINE OPR	M	031126	3340	3155	7200
19582	GROSSMAN, D. V.	2313	3340	HEAD	CHIFF, FAB. SECTION B	M	012629	3340	3510 3320 3540	9750
11130	COTATES, C. L.	2313	3340	EMPL	MACHINE OPR	M	061631	3340	3345 3360	8000
17590	GORTON, R. A.	2313	3340	EMPL	MACHINE OPR	M	032910	3340	3345 7320	8500
34040	GRFEN, S. D.	2313	3340	EMPL	MACHINE OPR	M	100621	3340	3360 7320	8000
19161	CHRISHMAN, C. H.	2313	3340	EMPL	MACHINE OPR	M	040630	3340	3345	7600
81135	MIRGINS, P. S.	2313	3340	EMPL	MACHINE OPR	M	080824	3340	3345	7200
15556	MCCONSON, P. O.	2314	3510	EMPL	ASSEMBLER	F	052123	3510	3340 3740	9500
11775	O-LINGER, D. D.	2314	3510	EMPL	ASSEMBLER	F	031136	3510	3745	5500
13467	OSTERBERG, W. T.	2314	3510	EMPL	ASSEMBLER	M	011325	3510	340	6500
23030	PASS, C. E.	2314	3510	EMPL	ASSEMBLER	F	100334	3510	3320 3570	8000
77524	PARRUS, J. G.	2314	3510	EMPL	ASSEMBLER	F	020230	3510	3320	7500
86024	RAIDWIN, R. C.	2320	0150	HEAD	MANAGER, SUPPLY GROUP	M	100116	0150	3740	10500
55396	DAVIS, J. M.	2321	3740	HEAD	CHIFF, STOCKROOM SECTION	F	040546	3740	3790	4000
55443	MORGAN, J. C.	2321	3745	EMPL	STOCK CLERK	M	080729	3745		8000
17784	MARTIN, V. O.	2321	3745	EMPL	STOCK CLERK	M	041340	3745		8500
42330	MOELLER, A. C.	2321	3745	EMPL	STOCK CLERK	M	110643	3745		5200
47573	TOMPSON, E. W.	2321	3745	EMPL	STOCK CLERK	M	062415	3745		6500
88453	NEWLIN, S. F.	2321	3745	EMPL	STOCK CLERK	M	030841	3745		5500
41555	DUSHAN, L. C.	2322	3740	HEAD	CHIFF, WAREHOUSE SECTION	M	112117	3740	3780	8000
34233	PELL, T. C.	2322	3745	EMPL	STOCK CLERK	M	082835	3745		6000
34823	MUELLE, L. D.	2322	3745	EMPL	STOCK CLERK	M	032145	3745		4000
76944	MYERS, L. M.	2322	3745	EMPL	STOCK CLERK	M	072344	3745		4000
92757	PULEN, R. A.	2340	3320	HEAD	MANAGER, SCHD. + DISP. GROUP	M	021730	3320	3450	14000
13060	PROVN, G. E.	2340	3320	EMPL	CLERK-TYPIST	F	100546	3320	3450	14500
20763	SWFEMAN, F. J.	2341	3320	HEAD	CHIFF, SCHEDULING SECTION	M	062626	3320		9000
20543	SALISBURY, T. C.	2341	3320	EMPL	SCHEDULER	M	081020	3320		7500
61157	SEABOIC, E. S.	2341	3320	EMPL	SCHFOULER	M	040735	3320		8100
22533	LUSTMAN, S. D.	2342	3550	HEAD	CHIFF, DISPATCHING SECTION	M	050324	3550		8000
11058	JOHNSON, C. P.	2342	3540	EMPL	DISPATCHER	M	071037	3540		7500
3604	JONES, L. E.	2342	3550	EMPL	EXPEDITER	M	121034	3550		7000
64974	KLEMER, R. J.	2342	3550	EMPL	EXPEDITER	M	092839	3550		6500
74093	MOOVER, E. J.	2342	3550	HEAD	MANAGER, PLANT ENGR. GROUP	M	090236	3550		9200
45192	MCWILLAN, C. M.	2350	7120	HEAD	CLERK-TYPIST	M	110515	7120	0150 3340 3510	13000
65473	POWLAND, F. R.	2351	7110	HEAD	CHIFF, PLANT MAINT. SECTION	F	051344	7110		4200
50499	DOXTATOR, M. H.	2351	7110	EMPL	MAINT ENGR (PL)	M	030920	7110	7145	9700
54966	JACOBSON, I. R.	2351	7110	EMPL	MAINT TECH (PL)	M	030920	7110	7115	9000
36756	ISAACS, R. A.	2351	7110	EMPL	MAINT TECH (PL)	M	101426	7110	7150	8200
17690	MUTTON, W. T.	2351	7310	EMPL	MAINT EQUIP. MAINT. SEC.	M	041822	7310	7160	7200
13090	FLAR, J. M.	2353	7120	HEAD	CHIFF, EQUIP. MAINT. SEC.	M	082924	7120	3140	10500
78885	LARNER, L. F.	2353	7120	EMPL	MAINT ENGR (E)	M	070433	7120	3340 3125	9800
67384	KARSTEN, P. E.	2353	7320	EMPL	MAINT TECH (E)	M	021326	7320	3340	8200
68754	KUMNS, C. F.	2353	7320	EMPL	MAINT TECH (E)	M	011236	7320	3140	8700
71191	ROACH, J. P.	2353	7320	EMPL	MAINT TECH (E)	M	030840	7320		7500

Attachment 11. Data Base: Organization (Sheet 1).

(ORG UNIT) ↓	(ORG NAME) ↓	(REPORTS TO) ↓	(QUANT) ↓	(AUTHORIZED SALARY) ↓	(BUDGET) ↓
2000	REPRESENTATIVE DIV. (JOB CODE) (UNIT CODE) (TITLE)	1000			
AUTHORIZED COMPLEMENT	0110 DIVISION MANAGER		1	28000	
	5210 SECRETARY		1	4000	
	2100 DEPARTMENT		1	510900	
	2300 DEPARTMENT		1	425800	970700
2100	DEVELOPMENT DEPT.	2000			
	1110 MANAGER		1	24000	
	5210 SECRETARY		1	5400	
	2110 GROUP		1	165300	
	2120 GROUP		1	109200	
	2130 GROUP		1	192000	
	2190 GROUP		1	27000	510900
2110	SYST. ENGR. GROUP	2100			
	1110 MANAGER		1	17500	
	5210 SECRETARY		1	4200	
	2111 SECTION		1	52600	
	2113 SECTION		1	49000	
	2115 SECTION		1	42000	165300
2111	PROPOSAL SECTION	2110			
	1110 CHIEF		1	14000	
	1120 MECH ENGR		1	10500	
	1130 ELEC ENGR		1	11500	
	1330 DRAFTSMAN		1	8500	
	1350 COST ESTIMATOR		1	8100	52600
2113	ADV. SYSTEMS SECTION	2110			
	1110 CHIEF		1	13000	
	1110 SYSTEMS ENGR		1	12000	
	1120 MECH ENGR		1	12000	
	1130 ELEC ENGR		1	12000	49000
2115	PROD. SPEC. SECTION	2110			
	1110 CHIEF		1	12000	
	1120 MECH ENGR		1	11000	
	1130 ELEC ENGR		1	11000	
	1330 DRAFTSMAN		1	8000	42000
2120	STDS. ENGR. GROUP	2100			
	1120 MANAGER		1	14500	
	5210 SECRETARY		1	4000	
	2122 SECTION		1	39200	
	2123 SECTION		1	44500	102200
2122	SYSTEM STDS. SECTION	2120			
	1120 CHIEF		1	11500	
	1120 MECH ENGR		1	9500	
	1130 ELEC ENGR		1	9700	
	1330 DRAFTSMAN		1	8500	39200

1 December 1965

3-27

TM-2624/100/00

Attachment 11. Data Base: Organization (Sheet 2).

----- 2123	COMPONENTY STDS. SECTION	2120			
	1120 CHIEF		1	11000	
	1120 MECH ENGR		1	9500	
	1130 ELEC ENGR		1	10000	
	1330 DRAFTSMAN		1	8500	
	9520 FILE CLERK		1	9500	44500
2130	COMPONENTY ENGR. GROUP	2100			
	1120 MANAGER		1	16500	
	5210 SECRETARY		1	4500	
	2131 SECTION		1	43000	
	2132 SECTION		1	41500	
	2133 SECTION		1	49000	
	2135 SECTION		1	37500	192000
2131	ENGR. SECTION A	2130			
	1120 CHIEF		1	12000	
	1120 MECH ENGR		2	22000	
	1330 DRAFTSMAN		1	9000	43000
2132	ENGR. SECTION B	2130			
	1130 CHIEF		1	11500	
	1130 ELEC ENGR		3	30000	41500
2133	PROD. ENGR. SECTION	2130			
	1150 CHIEF		1	11000	
	1150 PROD ENGR		2	20000	
	1330 DRAFTSMAN		1	8500	
	1365 TOOL DESIGNER		1	9500	49000
2135	MODEL SHOP SECTION	2130			
	3125 CHIEF		1	10000	
	1351 MECH TECHN		1	9000	
	1355 ELEC TECHN		1	9000	
	3125 TOOL MAKER		1	9500	37500
2190	CHIEF SCIENTIST	2100			
	1190 CHIEF SCIENTIST		1	22000	22000
2300	OPERATIONS DEPT.	2000			
	0130 MANAGER		1	22000	
	5210 SECRETARY		1	5000	
	2310 GROUP		1	163000	
	2320 GROUP		1	67500	
	2340 GROUP		1	72600	
	2350 GROUP		1	95700	425800

1 December 1965

3-28

TM-2624/100/00

Attachment 11. Data Base: Organization (Sheet 3).

2310	FACTORY OPNS. GROUP	2300			
	0150	MANAGER	1	14000	
	5210	SECRETARY	1	5000	
	2311	SECTION	1	50000	
	2313	SECTION	1	50000	
	2314	SECTION	1	44000	163000
2311	FAB. SECTION A	2310			
	3340	CHIEF	1	10000	
	3340	MACHINE OPR	5	40000	50000
2313	FAB. SECTION B	2310			
	3340	CHIEF	1	10000	
	3340	MACHINE OPR	5	40000	50000
2314	ASSEMBLY SECTION	2310			
	3510	CHIEF	1	9500	
	3510	ASSEMBLER	5	34500	44000
2320	SUPPLY GROUP	2300			
	0150	MANAGER	1	10500	
	5220	CLERK-TYPIST	1	4000	
	2321	SECTION	1	30000	
	2322	SECTION	1	23000	67500
2321	STOCKROOM SECTION	2320			
	3740	CHIEF	1	8000	
	3745	STOCK CLERK	4	22000	30000
2322	WAREHOUSE SECTION	2320			
	3740	CHIEF	1	8000	
	3745	STOCK CLERK	3	15000	23000
2340	SCHED. + DISP. GROUP	2300			
	3320	MANAGER	1	11000	
	5220	CLERK-TYPIST	1	4500	
	2341	SECTION	1	24600	
	2342	SECTION	1	32500	72600
2341	SCHEDULING SECTION	2340			
	3320	CHIEF	1	9000	
	3320	SCHEDULER	2	15600	24600
2342	DISPATCHING SECTION	2340			
	3540	CHIEF	1	8000	
	3540	DISPATCHER	2	13000	
	3550	EXPEDITER	2	11500	32500

1 December 1965

3-29  
(Page 3-30 blank)

TM-2624/100/00

Attachment 11. Data Base: Organization (Sheet 4).

2350	PLANT ENGR. GROUP	2300			
	7120	MANAGER	1	13500	
	9220	CLERK-TYPIST	1	4200	
	2351	SECTION	1	34500	
	2353	SECTION	1	44000	95700
2351	PLANT MAINT. SECTION	2350			
	7110	CHIEF	1	10000	
	7110	MAINT ENGR(PL)	1	9000	
	7310	MAINT TECH(PL)	2	15500	34500
2353	EQUIP. MAINT. SECTION	2350			
	7120	CHIEF	1	10000	
	7120	MAINT ENGR (E)	1	9500	
	7320	MAINT TECH (E)	3	24500	44000

## THREE COLINGO-LIKE APPROACHES TO THE DATA BASE PROBLEM

Presented by: Frank Cataldo  
The MITRE Corporation

INTRODUCTION

The three approaches illustrated here differ so much because of the differences in the three systems used--ADAM, COLINGO D, and COLINGO C-10. Due to limitations in COLINGO D, finding a solution was difficult and dependent on operator intervention. To choose a solution in ADAM and in COLINGO C-10 was difficult because of the number of possibilities.

All three systems are on-line to the user and operate on self-descriptive data bases. COLINGO D is a production system on a small machine with emphasis on tapes. ADAM is a complex and flexible experimental system on a large machine with emphasis on random access to disc and multiple simultaneous users. It has provisions for extensive computing, as in scheduling applications. COLINGO C-10 is also random-access oriented, and suitable for a medium size machine. It is a prototype design, which is expected to lead to a design for production. Both ADAM and COLINGO C-10 provide for displays, use a multilevel-tree, variable-length entry file structure, and have cross-file capability. In contrast, COLINGO D has a more rigid structure, and is essentially one-file-at-a-time. All three systems allow programming at the query level through stored queries equivalent to subroutines and macros at the query level, with the power increasing from COLINGO D, through ADAM, to COLINGO C-10.

The character of the systems is dictated, in part by the level at which they can be programmed or at which they are interpretive. COLINGO D is high-level only. Therefore, it is simpler but more restricted and less open-ended. ADAM can be programmed at several levels, and is interpretive mainly at an intermediate level. At present, COLINGO C-10 is very regular and interpretive mostly at a lower level, and is therefore more flexible and easiest to fully understand and use. As a consequence, it is also more wordy. A design for a preprocessor has been implemented to interface at the user level or to abbreviate for well-structured applications; but has proved too slow for normal on-line use in its present form. It has not been used at all in the examples. The preprocessor and the ADAM translator allow for extensive front-end changes on a user-by-user basis, if needed.

Only the COLINGO D version was run all the way on the computer.

The COLINGO C-10 example is based on the work of C. S. Wells, G. D. Steil and L. R. Bennis; the ADAM example on that of J. C. Clapp, R. G. Curtis and T. L. Connors.

## 1. COLINGO D

### GENERAL DESCRIPTION OF COLINGO D

The first approach to solving the data base problem was to apply COLINGO D (hereafter called COLINGO), a data management system which has been in operation for the past several years. Since this is basically a single-file system whose file structure is organized around a restricted number of structural levels, the problem was modified in some respects to conform to the capabilities of COLINGO. In general, most of this tailoring was performed during the file-generation phase and is noted throughout the document where it occurs.

The major features of COLINGO are a user-oriented query language coupled with an on-line input capability. The file structure of the system is COBOL-oriented and makes use of a fixed-format, data-division concept. Information about a single object or entity is contained within a single data record called a "Master" record. However, one level of repetitive dependent information is allowed in the form of "trailers."

A file record is segmented into fixed fields by the file "Dictionary," each field being associated with a "Property Name." Levels of property names are allowed; for example, a property called SKILL might contain the property's name and code. The name can be referenced by either NAME or SKILL/NAME. Should one want both properties, the reference SKILL will suffice. For example:

#### PRINT SKILL

will actually print two quantities: NAME and CODE. With this as a guideline, the files were organized as described below.

### FILE ORGANIZATION

#### Personnel File

The Personnel File was organized as having one master record per employee, with associated trailers for the repeated fields of skill code and skill name. This gives us the facility to add or delete skills. It also gives us the capability to qualify an object on a particular skill or combination of skills, or any combination of single-valued and multiple-valued properties.

#### Organization File

In view of the reports that require the combined Personnel and Organization information, and the fact that COLINGO was primarily designed to produce demand reports, and has very limited cross-file referencing capability, the Organization



file was developed on a unit basis with one master record per unit and associated trailers for the repeated groups containing the authorized information. Additional trailers were added to each Section (unit) containing current actual staffing. This was done by passing the Personnel file and extracting the same fields on a Section basis as were contained in the repeated groups for each Section. Therefore, the modified Organization file has "Actual" Quantity and Salary information included within the Section records. The Group and Department records contain Actual Quantity and Salary information wherever people (Mgr and Secretary) are included and have the same structure as a Section record, with slots available for Actual totals.

This will allow us to have the combined information available for easy production of any of the desired reports. It does mean, however, that any personnel transaction such as salary changes, accessions, or terminations, etc., that affect budget figures must be reflected in this file. In other words, some of the time we will have to update both files to reflect changes.

#### FILE GENERATION

The inputs to the FILE GENERATION program consisted of control cards and the data records on cards. A sample of the input for creating the Personnel file is shown on Figure 1-1. The first card, called an Identification card, is used to name the file to be created. This file name (PER FILE) is inserted in the header record of the data file produced. The next card, a File Description card, is used to describe the parameters needed by the FILE GENERATION program. Such elements as type of input, master-trailer option, ID field, FORMAT-NO, length of input record, length of output record, etc., are entered.

A brief description of two of these elements, ID and FORMAT-NO, is essential to understand FILE GENERATION. In our Personnel file, the employee number is a principal field whose value is unique to a data record, which was designated as the ID field. During the file-generation process, the ID field is examined and a change of value results in the generated record being identified as a master record. A repetition of the previous ID value, or all blanks in the field, results in the record being identified as a trailer record.

The data for the Personnel file ran over one card per master record. To combine this information during File Generation to produce one COLINGO data record, a field within each record is used to identify that record as to its relative location within the COLINGO data record. This field is referred to as FORMAT-NO. For the Personnel data, the number "1" was punched in column 1 of the first card of each record as FORMAT-NO. The data punched on the first card included EMPL-NO through Date of Birth. The second card of the record had the number "2" punched in column 1 as FORMAT-NO. Data punched on the second card included skill, salary, and skill name, which we chose to include in the file. Also included was a field called NO-OF-SKILLS, since COLINGO cannot count trailers (repetitions) easily. As many 2-cards as necessary to describe an employee



followed, each containing one skill and its associated skill name (i.e., trailers).

The COLINGO data record is written with a maximum data record length of 999 characters. When data records are less than 500 characters, they will be blocked as many times as possible to fit within a 999-character physical record. Since the input for a master record was on two cards, the length of a logical record was stated as 160 characters; therefore, 6 records are contained within one physical record. The actual structure of the record for the man named Carr is shown in Figure 1-2. Since he had 3 additional skills, the number of logical records stored for man Carr was 4 (640 characters).

The Organization data (Figure 1-3) received the same treatment, having two FORMAT-NO's per record, each punched in column 80, one for the Authorized data and one for the Actual data. The field called BUDGET in the original data was not punched on the input cards. Since this total is subject to change, we chose to compute it upon request.

The "END" card tells FILE GENERATION that no more data follows. The "NO MORE FILES" card terminates the File Generation program. This was used, since COLINGO can generate successive files during the same computer run. Again, since the input for a master record was on two cards, the size of a logical record was also 160 characters. The number of logical records necessary for Section 2111 was 11 (1760 characters). The actual structure of Section 2111 is shown in Figure 1-4.

#### DICTIONARY GENERATION

Along with the data files, it is necessary to generate a corresponding dictionary. This function is performed by the Dictionary Generation (DIG) program. The inputs to DIG are COBOL Data Divisions. A Data Division contains all of the information necessary to completely describe a data file. This information includes the name of the data file (line beginning with FD), the name, type and length of data fields, control fields, and sub-field (lines beginning with 02 and 03).

The Data Divisions are input to DIG via punched cards. The program extracts and condenses this information, calculates the relative locations of the data fields, and writes all of this information on magnetic tape. Figures 1-5 and 1-6 show the Data Divisions used to create the Personnel file (PER-FILE) and the Organization file (ORG-FILE). The dictionaries (from one tape) and the files (on another tape) were then combined on one tape in the following sequence: Personnel file (Dictionary and Data) followed by Organization file (Dictionary and Data), for use during our processing. Combining files and dictionaries on one tape is not required. COLINGO could have handled these from separate tape drives but combining was done for ease of tape handling at our facility.

1 December 1965

3-36

TM-2624/100/00

MASTER	ID/EMPL-NO.	NAME	UNIT-CODE	SKILL	SALARY	SKILL-NAME
TRAILER	57060	CARR, P.L.	2100	1110	24000	SYSTEMS ENGR
TRAILER				1130		ELEC ENGR
TRAILER				1135		COMMU ENGR
TRAILER				0130		ADMIN DEPT
TRAILER						

Figure 1-2. Personnel File--Typical Data Record



	ID/ORG CODE/JOB	ORG/TITLE	REPORTS-TO	AUTHORIZE		ACTUAL
				QUAN	SALARY	QUAN
MASTER	2111	PROPOSAL SECTION	2110			
TRAILER	1110	CHIEF		1	14000	
TRAILER	1120	MECH ENGR		1	10500	
TRAILER	-	-		-		
TRAILER	-	-		-		
TRAILER	1110	CHIEF				1
TRAILER	1120	MECH ENGR				1
TRAILER	-	-		-		
TRAILER	-	-		-		

Figure 1-4. Organization File--Typical Data Record

FD	PER-FILE	
02	DISK - ID	PICTURE IS X (14).
02	TRAILER	PICTURE IS X (1).
02	FORMAT - NO - 1	PICTURE IS X (1).
02	ID	PICTURE IS X (5).
	03 EMPL - NO	PICTURE IS 9 (5).
02	NAME	PICTURE IS X (19).
02	UNIT - CODE	PICTURE IS 9 (4).
<hr/>		
02	SKILL	PICTURE IS 9 (4).
02	FILLER	PICTURE IS X (22).
02	SALARY	PICTURE IS 9 (7).
02	FILLER	PICTURE IS X (2).
02	NO - OF - SKILLS	PICTURE IS 9 (1).
02	SKILL - NAME	PICTURE IS X (25).
02	FORMAT - NO - 2	PICTURE IS X (1).

---

Figure 1-5. Data Divisions--Personnel File

FD	ORG-FILE	
02	DISK - ID	PICTURE IS X ( 14 ).
02	TRAILER	PICTURE IS X ( 1 ).
02	ID	PICTURE IS X ( 4 ).
	03 ORG	PICTURE IS 9 ( 4 ).
02	CODE	PICTURE IS X ( 8 ).
	03 JOB	PICTURE IS 9 ( 4 ).
	03 UNIT	PICTURE IS 9 ( 4 ).
02	ORG	PICTURE IS X ( 20 ).
	03 TITLE	PICTURE IS X ( 20 ).
02	REPORTS-TO	PICTURE IS 9 ( 4 ).
02	FILLER	PICTURE IS X ( 5 ).
02	AUTHORIZE	PICTURE IS X ( 30 ).
	03 QUAN	PICTURE IS 9 ( 15 ).
	03 SALARY	PICTURE IS 9 ( 15 ).
02	FILLER	PICTURE IS X ( 8 ).
02	FORMAT-NO-1	PICTURE IS X ( 1 ).




Figure 1-6. Data Divisions--Organization File



### Control Reports

The statements used to produce the control reports are shown in Figure 1-7, and were executed in the sequence as shown. Due to the structure of our files and the nature of the report, it required only a simple count of objects (master records) in both files plus one vertical sum of one property (SALARY) in the Personnel File. Since COUNT and SUM are output directly to the system console, COMMENT statements (A, B, and E) were used to give meaning to the answers. The COMMENT verb will print whatever follows within a set of quotation marks on the system console. The files were each passed once through the system to produce the answers. The number of program modules activated by the system (accessed from disc, brought into core and operated) was 10, shown as underlined verbs on Figure 1-7. Each module is activated sequentially as it is encountered in a statement.

### Updating Transactions

For each updating of the file, a separate query is created. We have taken the specific problem and generated the three queries necessary to perform these transactions. The statements required are shown in Figure 1-8.

In the first case (the termination), it was necessary to delete an object from the file. This was done by writing the entire file over again without the object in question. Another possibility would have been to create a property called TERMINATED to use as an indicator. However, this would eventually cause the file to grow out of proportion to its data content.

To change a salary, the second statement was used. It will be noted that the salary is contained in the master record for each individual while the trailers contained incidental information. It was therefore necessary to use the (03) level modifier in the query to prevent each of the trailers from being modified.

In adding a skill code, we are qualifying and modifying trailer information and therefore the (03) modifier was again used so that only the specific trailer in question is changed. Since we also chose to carry the skill name in the file rather than in a separate file, this property was also updated.

### DATA AND PROGRAM FLOW

COLINGO action verbs in an input message are processed sequentially, with intermediate and final results being output to tape. Most action verbs pass an entire data file and produce another self-describing file before the next action verb is called in by the Executive routine.

A COMMENT 'FIELD 1 IS NUMBER OF PERSONS EMPLOYED'.  
 B COMMENT 'FIELD 2 IS TOTAL ACTUAL ANNUAL SALARIES'.  
 C GET PER-FILE IF 1 EQ 1 TYPEOUT COUNT COMPUTE / BLANK/SUM/5 SALARY  
 D COMMENT 'FIELD BELOW IS NUMBER OF ORGANIZATION UNITS'.  
 E GET ORG-FILE IF 1 EQ 1 TYPEOUT COUNT

Figure 1-7. Control Reports

A GET PER-FILE IF ID NQ 33194 WRITE/BLANK/3 ALL.  
 B GET FILE/3 IF ID (03) EQ 91152 CHANGE/BLANK/4 SALARY TO 8500.  
 C GET FILE/4 IF ID EQ 85657 AND SKILL (03) EQ BLANKS CHANGE/BLANK/3  
 SKILL TO 3750 SKILL-NAME TO DISPATCH+ASST.

Figure 1-8. Updating Transactions

Figure 1-9 lets us examine this flow as applied to a previous query, statement B of Figure 1-8: GET FILE/3 IF ID (03) EQ 91152 CHANGE/BLANK/4 SALARY TO 8500.

The first action verb to execute is the GET verb, which subsets the data base to locate the PER-FILE (on tape drive 3, due to the previous message). The next action verb to work is IF, which tags any data record (in this case, one) containing ID of 91152. This produces a new file on another tape drive with all the records of the original PER-FILE, but only the record that qualified being tagged. Finally, the CHANGE verb operates, updating the tagged record and simply copying the others.

The next section on Control Reports illustrates this more thoroughly in the context of the specific problem.

#### Periodic Report

To generate the periodic reports, it was decided to produce a "stored query" which could be called simply by performing an EXECUTE statement followed by the name of the stored query SECTION REPORT or GROUP REPORT, thus not requiring the operator to type the entire procedure.

The COLINGO Report Generator could have been used to produce data in the specified format as well as the necessary totals (except for the deviations). However, the group report and the later reports were dependent on certain totals which could be obtained during the production of the periodic report; therefore, it was decided to compute and store these totals in the file rather than use the Report Generator to compute them, thus losing them for future use. This method is slower for this report than using the Report Generator method but makes the overall problem faster and, in fact, makes the Exception and Hypothesis reports possible.

One further bit of explanation is required to describe the reason it is necessary to cycle through the organization file generating a deck of punched cards. COLINGO has a restriction that only one computation is allowed per pass of the file. Each pass of the file constitutes an independent query. Linkage can be made between queries only through the HOLD capability. There is a limitation on HOLD, in that only ten parameters may be held at any one time. It was therefore decided to generate a deck of cards containing the identification of each group or section which would then be placed in the card reader by a computer operator. Each card is read in turn by query A in Figure 1-10 and its value "held" for processing by the remaining queries (B through J).

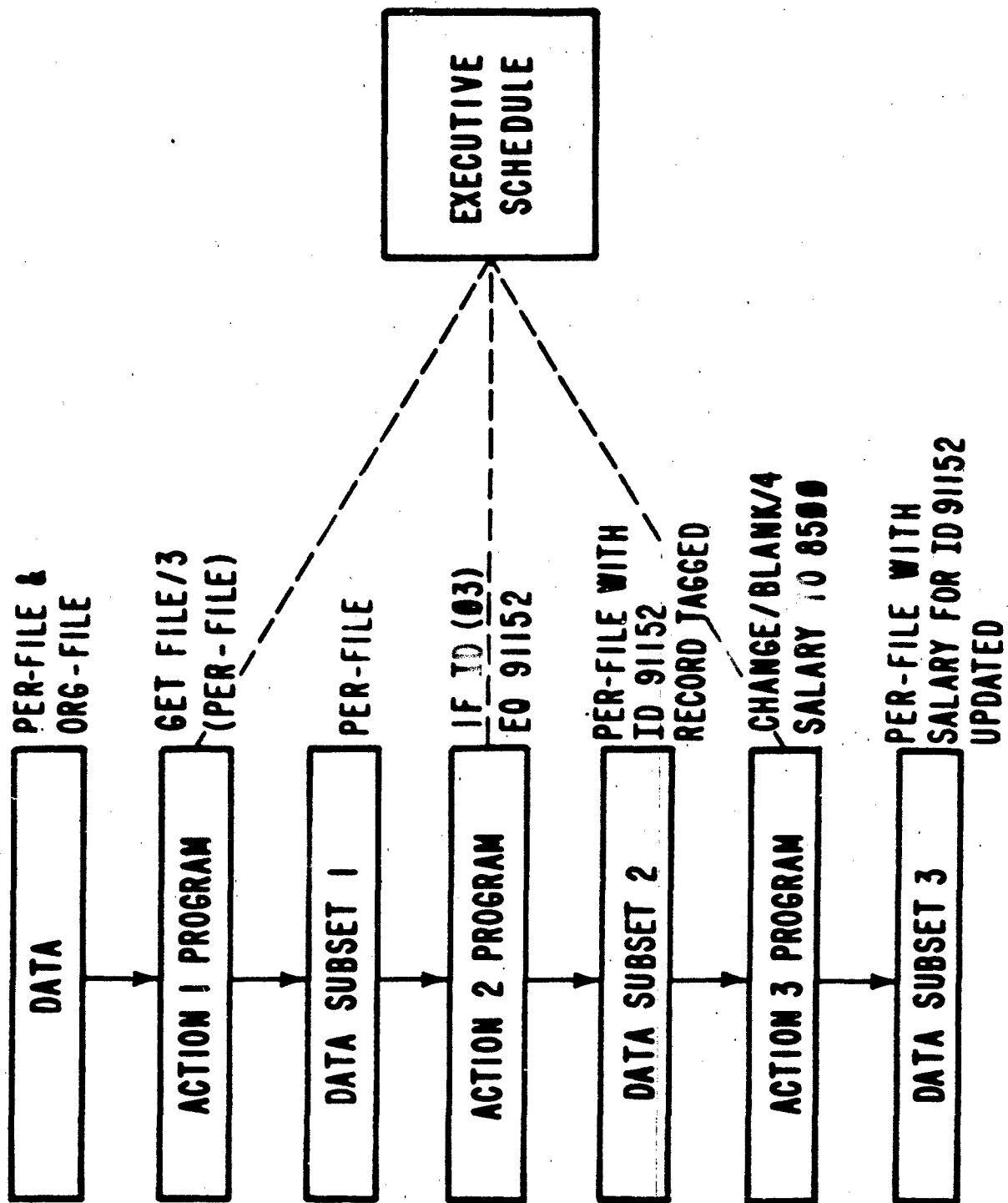


Figure 1-9. Data and Program Flow

SECTION REPORT GET ORG-FILE IF ID (+4,-1) NO 0 PUNCH ID COMMENT 'INSERT PUNCHED CARDS IN  
READER' PAUSE EXECUTE A.

A GET ORG-FILE IF ID EQ -C- HOLD ID EXECUTE B

B GET ORG-FILE IF ID EQ HOLD0/ID COMPUTE/BLANK/4 AUG -AUTHORIZE/QUAN +X EXECUTE C.

C GET FILE/4 IF ID EQ HOLD0/ID COMPUTE/BLANK/5 AUS -AUTHORIZE/SALARY +X EXECUTE D.

D GET FILE/5 IF ID EQ HOLD0/ID COMPUTE/BLANK/4 ACQ -ACTUAL/QUAN +X EXECUTE E.

E GET FILE/4 IF ID EQ HOLD0/ID COMPUTE/BLANK/5 ACS -ACTUAL/SALARY +X EXECUTE F.

F GET FILE/5 IF ID EQ HOLD0/ID COMPUTE/BLANK/4 BD -ACS - AUS EXECUTE G.

G GET FILE/4 IF ORG/TITLE (+10,-4) EQ HOLD0/ID CHANGE/BLANK/5 AUTHORIZE/QUAN TO AUG  
AUTHORIZE/SALARY TO AUS ACTUAL/QUAN TO ACQ ACTUAL/SALARY TO ACS BUDGET/DEV  
TO BD EXECUTE H.

H GET FILE/5 IF ID EQ -C- HOLD ID EXECUTE J IF/NOT COMMENT 'ALL DONE EXECUTE K FOR  
SECTION REPORT, GF FOR GROUP REPORT'.

J GET FILE/5 IF ID EQ HOLD0/ID COMPUTE/BLANK/4 AUG -AUTHORIZE/QUAN +X EXECUTE C.

K GET FILE/5 IF ID (+4,-1) NO 0 SORT (ASC) ID TITLE IS 'SECTION REPORT' PRINT  
(HEADINGS) ID CODE/JOB REPORTS-TO AUTHORIZE ACTUAL BUDGET.

Figure 1-10. Section Report

### Section Report

Although the procedure described above seems somewhat cumbersome, the entire program can be given to a computer operator to perform and his actions are clearly defined by the typed comments. In addition, a byproduct in flexibility is obtained from this procedure, in that individual group or section reports can be performed for specific groups or sections merely by punching an identification card and executing Statement A rather than SECTION REPORT.

The "-C-" of statement A tells COLINGO that parameter insertion via card input is requested to complete this statement. A card will be read and the punched value inserted at this point into the statement. HOLD stores the qualifying ID in a temporary file labeled HOLD.

The various sums are computed, the deviation is formed, and the results are transferred via the CHANGE verb of the statement labeled G into the properly labeled properties. Each of these statements passes the file once and writes an updated version on a scratch tape (either 4 or 5). Note that if COLINGO could perform multiple computes per pass, statements B, C, D and F could have been combined, with consequent time saving.

The program will be cycled once per punched-card input, ending when the card reader is empty as shown by statement J. The verb IF/NOT acting as a conditional branch will cause the comments to be printed on the system console. Since this program will be used to produce the Group Report also, the operator will have to insert the correct EXECUTE statement as noted to produce the actual report. When this comment from statement J is printed out, COLINGO will return control of the computer back to the system operator.

This is the most time-consuming portion of the total problem, since it forms the basis for all of the later reports. One file pass is made for each statement A through K, and the total cycle is iterated for each section or group in the file. The total time involved is a function of the file size. However, the later reports (Exception and Hypothesis) are performed with one pass through this modified file.

A sample output of the Section Report is shown in Figure 1-11.

### Group Report

Producing the Group Report required a "front end" series of statements as shown in Figure 1-12. This "front-end" updates the Actual Section totals contained in each Group from the Section totals produced by the Section Report program. Then it punches a card for each Group containing its ID as shown in statement GE and branches to statement A of the SECTION REPORT program to perform the various sums and updates.

1 December 1965

3-47

TM-2624/100/00

## SECTION REPORT

IO/JRY	CODE/JUR	ORG/TITLE	REPORTS-TO	AUTHORIZE/QUAN	AUTHORIZE/SALARY	ACTUAL/QUAN	ACTUAL/SALARY	JUDGE 1/11EV
2111		PROPOSAL SECTION	2110					
	1110	CHIEF		1	14000			
	1120	MECH ENGR		1	10500			
	1130	ELEC ENGR		1	11500			
	1140	DRAFTSMAN		1	8500			
	1150	CUST ESTIMATOR		1	8100			
	1110	CHIEF				1	14000	
	1120	MECH ENGR				1	10500	
	1130	ELEC ENGR				1	11500	
	1140	DRAFTSMAN				1	8500	
	1150	CUST ESTIMATOR				1	8100	
	TOTAL-2111			5	52600	5	52600	0
2113		ADV SYS SECTION	2110					
	1110	CHIEF		1	13000			
	1120	MECH ENGR		1	12000			
	1130	ELEC ENGR		1	12000			
	1110	CHIEF				1	13000	
	1120	MECH ENGR				1	12000	
	1130	ELEC ENGR				1	12000	
	TOTAL-2113			4	49000	4	48000	-1000
2115		PROD SPEC SECTION	2110					
	1110	CHIEF		1	12000			
	1120	MECH ENGR		1	11000			
	1130	ELEC ENGR		1	11000			
	1140	DRAFTSMAN		1	8000			
	1110	CHIEF				1	11000	
	1120	MECH ENGR				1	10500	
	1130	ELEC ENGR				1	11000	
	1140	DRAFTSMAN				1	8000	
	TOTAL-2115			4	42000	5	51500	9500

Figure 1-11. Section Report

GROUP REPORT	COMMENT 'INSERT PUNCHED CARDS FROM SECTION REPORT IN READER' PAUSE EXECUTE GA.
GA	GET FILE/5 IF CODE/UNIT EQ -C- HOLD ID ACTUAL/SALARY EXECUTE GB IF/NOT COMMENT 'ALL DONE UPDATED FILE ON TD4 RESELECT AS TD2' EXECUTE GE.
GB	GET FILE/5 IF CODE/UNIT EQ HOLD0/ID AND AUTHORIZE/SALARY EQ BLANKS CHANGE/BLANK/4 ACTUAL/SALARY TO HOLD0/ACTUAL/SALARY EXECUTE GC.
GC	GET FILE/4 IF CODE/UNIT EQ -C- HOLD ID ACTUAL/SALARY EXECUTE GD IF/NOT COMMENT 'ALL DONE UPDATED FILE ON TD5 RESELECT AS TD2' EXECUTE GE
GD	GET FILE/4 IF CODE/UNIT EQ HOLD0/ID AND AUTHORIZE/SALARY EQ BLANKS CHANGE/BLANK/5 ACTUAL/SALARY TO HOLD0/ACTUAL/SALARY EXECUTE GA.
GE	GET ORG- FILE IF ID(+3,-1) NQ 0 PUNCH ID COMMENT 'INSERT PUNCHED CARDS IN READER' PAUSE EXECUTE A.
GF	GET FILE/5 IF ID(+3,-1) NQ 0 SORT (ASC) ID TITLE IS 'GROUP REPORT' PRINT (HEADINGS) ID CODE ORG REPORTS-TO AUTHORIZE ACTUAL BUDGET.

Figure 1-12. Group Report



The comments regarding time consumed made in the paragraph on Section Reports are applicable as well to the group report; i.e., this information is placed in the file for later use by the Exception and Hypothesis reports.

Upon completion of all passes, the operator will enter EXECUTE GF as noted in the console cue to produce the Group Report, a sample of which is shown in Figure 1-13.

#### Demand Reports

Each of these reports required two passes of the Personnel file, one to qualify and one to sort. The statement used to select the qualifying records for the first report is shown in Figure 1-14. This was again a combination qualification on master and trailer per object. Dictionary re-definition was required on the field called DOB (birthdate) which was originally stored as a six-digit field. Our interest was in the year subset of the property DOB. The (+6, -2) notation allows us to redefine the field limits of this property for this query to only look at year.

The output produced as a result of this query is shown in Figure 1-15. The standard COLINGO output produces vertical columns of data values, with automatic horizontal overflow and justification procedures. The sub-directive (HEADINGS) forces COLINGO to label each column of values with the actual property name. Normally, a key is printed first relating a property name to a vertical column. This allows data to be packed more tightly, producing more columns per 132-column horizontal page.

The statement used to select the qualifying records for the second report is shown in Figure 1-16. A combined AND/OR operation was performed on the field SKILL (skill code). The output produced is shown in Figure 1-17.

#### Exception Report

After the program (series of CCL statements) that had to be developed to produce the Section and Group Reports, the production of the Exception Report is a simple query of the updated file just produced. Since the updated file also carries along its own dictionary (the ORG-FILE dictionary), it can be referenced by the name of ORG-FILE.

The query is shown in Figure 1-18. It is a combination column-row qualification to select only Sections and within Sections to qualify on the trailer record containing the Total information. A sample printout is shown in Figure 1-19.

1 December 1965

3-50

TM-2624/100/00

GROUP REPORT

10/ORG CODE/JOB CODE/UNIT ORG/TITLE	REPORTS-TO	AUTHORIZE/CAN	AUTHORIZE/SALARY	ACTUAL/QUAN	ACTUAL/SALARY	BUDGET/DEV
2110						
1110	SYS ENGR GROUP	2100				
5210	MANAGER		1	17500		
	SECRETARY		1	4200		
	2111 SECTION		1	52500		
	2113 SECTION		1	49000		
	2115 SECTION		1	42000		
1110	MANAGER		1		17500	
5210	SECRETARY				4200	
	2111 SECTION		1		52600	
	2113 SECTION		1		48000	
	2115 SECTION		1		51500	
	TOTAL-2110		5	165300	173800	8500

Figure 1-13. Group Report

GET PER-FILE

IF SKILL EQ 3340 AND NO-OF-SKILLS GR 2 AND SEX EQ M

AND LEVEL NQ HEAD AND DOB (+6,-2) GR 16 AND LS 35

SORT (ASC) NAME

PRINT (HEADINGS) NAME ID UNIT-CODE SKILL SKILL-NAME.

Figure 1-14. Demand Report--Query, First Report

UNCLASSIFIED

07SEP65

NAME	ID/EMPL-NO	UNIT-CODE	SKILL	SKILL-NAME
BOYD, W. V.	15052	2135	1351	MECH TECHN
			3340	MACH OPR
			7320	MAINT TECH E
			3370	PAINT OPR
CUATES, C. L.	81130	2313	3340	MACH OPR
			3345	MILL MACH OPR
			3360	HEAT TREAT OPR
FLETCHER, M. W.	21675	2133	1365	TOOL DESGN
			3125	TOOL MAKER
			3340	MACH OPR
			7320	MAINT TECH E
GORTON, R. A.	17590	2313	3340	MACH OPR
			3345	MILL MACH OPR
			7320	MAINT TECH E
GREEN, S. D.	34840	2313	3340	MACH OPR
			3360	HEAT TREAT OPR
			7320	MAINT TECH E
LARNER, L. F.	78885	2353	7120	MAINT ENGR E
			3340	MACH OPR
			3125	TOOL MAKER
LEVITT, P. S.	32924	2311	3340	MACH OPR
			3345	MILL MACH OPR
			3360	HEAT TREAT OPR
LITTLE, E. F.	42055	2311	3340	MACH OPR
			3360	HEAT TREAT OPR
			3550	EXPEDITER
MILLEN, F. L.	92467	2311	3340	MACH OPR
			3510	ASSEMBLER
			7320	MAINT TECH E
			3570	DISPATCH ASS

Figure 1-15. Demand Report--Output, First Report

GET PER-FILE

IF SEX EQ M AND LEVEL NO HEAD AND

SKILL EQ 1110 OR GR 1129 AND LS 1140

SORT (ASC) NAME

PRINT (HEADINGS) NAME ID UNIT-CODE SKILL SKILL-NAME.

Figure 1-16. Demand Report, Query, Second Report

09SEP65

UNCLASSIFIED

NAME	ID/EMPL-NO	UNIT-CODE	SKILL	SKILL-NAME
APGAR, A. J.	H2802	2115	1130	ELFC ENGR
ARNETTE, L. J.	37114	2115	1130	ELFC ENGR
BITTINGER, G. J.	13581	2113	1143	ELFC ENGR
FRANCIS, G. C.	21777	2113	1110	SYSTEMS ENGR
			1130	ELFC ENGR
HENDERSON, R. G.	18130	2111	1130	ELFC ENGR
MASANO, L. M.	36472	2132	1130	ELFC ENGR
RAYMOND, M. F.	28654	2123	1130	ELFC ENGR
SILCOFF, D. M.	10295	2132	1130	ELFC ENGR
SKINNER, J. P.	40876	2132	1130	ELFC ENGR
STADENMAN, P. K.	40923	2122	1130	ELFC ENGR

Figure 1-17. Demand Report--Output, Second Report

GET ORG-FILE

IF ID (+4,-1) NQ Ø AND CODE/JOB EQ BLANKS  
AND ACTUAL/QUAN GQ AUTHORIZE/QUAN  
AND BUDGET/DEV GQ 800 OR LQ -800

SORT (ASC) ID

TITLE IS 'EXCEPTION REPORT'

PRINT (HEADINGS) ID CODE/JOB REPORTS-TO  
AUTHORIZE ACTUAL BUDGET.

Figure 1-18. Exception Report, Query

1 December 1965

3-55

TM-2624/100/00

09SEP65

EXCEPTION REPORT

ID/ORG	CODE/JOB	ORG/TITLE	REPORTS-TO	AUTHORIZE/QUAN	AUTHORIZE/SALARY	ACTUAL/QUAN	ACTUAL/SALARY	MIN.E.I./DEV
2113	ADV SYS SECTION							
	1110	CHIEF	2110	1	13000			
	1110	SYSTEMS ENGR		1	12000			
	1120	MCH ENGR		1	12000			
	1130	ELEC ENGR		1	12000			
	1110	CHIEF				1	13000	
	1110	SYSTEMS ENGR				1	12000	
	1120	MCH ENGR				1	12000	
	1130	ELEC ENGR				1	11000	
		TOTAL-2113		4	49000	4	48000	-1000
2115	PRJD SPCL SECTION							
	1110	CHIEF	2110	1	12000			
	1120	MCH ENGR		1	11000			
	1130	ELEC ENGR		1	11000			
	1130	DRATSMAN		1	8000			
	1110	CHIEF				1	11000	
	1120	MCH ENGR				1	11000	
	1130	ELEC ENGR				1	10500	
	1130	ELEC ENGR				1	11000	
	1130	DRATSMAN				1	8000	
		TOTAL-2115		4	42000	5	51500	9500

Figure 1-19. Exception Report, Sample Printout

### Hypothesis Report

To produce the Hypothesis Report required the series of statements shown in Figure 1-20 plus iterations through the previously described programs labeled SECTION REPORT and GROUP REPORT. We did not iterate through these programs to actually produce the report. What we did was to bring the file up to date to reflect the proposed changes.

The statement labeled HA scans the file for Department 21 only and within each qualifying record for CODE/JOB of draftsman. A vertical sum is computed of Authorized Salary and number of draftsmen. Statement HB requires parameter insertion via the system console "-1-, -2-". The output of statement HA is entered to satisfy the variables and section 2123 is updated to contain all the authorized draftsmen. Statement HC removes all other authorized draftsmen from their respective units. Statement HD produces a printout of the file now ready for operation by programs SECTION REPORT and GROUP REPORT. A sample of the output is shown in Figure 1-21.

### Alternate Hypothesis Method

An alternate method to produce this modified report is shown in Figure 1-22. The operator need only activate the statement EXECUTE HYPO. The COMMENTS will be printed on the system console for operator notification. Then the statement HE will be executed. HE is an EXECUTE statement that contains a series of message labels which will be chained and operated in the sequence stated, under control of the EXECUTIVE program. Defining the order of message labels in an EXECUTE statement gives one a chance to execute a program in a variety of sequences, producing a kind of on-line test and evaluate function.



HA GET ORG - FILE IF ID (+2) EQ 21 AND CODE/JOB EQ 1330 AND ACTUAL/QUAN EQ BANKS  
COMPUTE/BLANK/SUM/5 AUTHORIZE/SALARY TYPEOUT COUNT.

HB GET ORG - FILE IF ID EQ 2123 AND CODE/JOB EQ 1330 AND ACTUAL/SALARY EQ BLANKS  
CHANGE/BLANK/5 AUTHORIZE/QUAN TO -1-. AUTHORIZE/SALARY TO -2-.

HC GET FILE/5 IF ID (+2) EQ 21 AND ID NQ 2123 AND CODE/JOB EQ 1330 AND ACTUAL/QUAN  
EQ BLANKS CHANGE/BLANK/3 CODE/JOB TO BLANKS ORG/TITLE TO BLANKS  
AUTHORIZE/QUAN TO BLANKS AUTHORIZE/SALARY TO BLANKS.

HD GET FILE/3 IF ID EQ 21 SORT (ASC) ID TITLE IS 'HYPO REPORT' PRINT (HEADINGS)  
ID CODE ORG REPORTS - TO AUTHORIZE.

Figure 1-20. Hypothesis Report

1 December 1965

3-58

TM-2624/100/00

ID/ORG	CODE/JOB	CODE/UNIT	ORG/TITLE	REPORTS-TO	AUTHORIZE/QUAN	AUTHORIZE/SALARY
<hr/>						
	1120		MECH ENGR			
	1130		ELEC ENGR			
			TOTAL-2122			
2123			COMPONENTS STDS SEC	2120		
	1120		CHIEF		1	11000
	1120		MECH ENGR		1	9500
	1130		ELEC ENGR		1	10000
	1330		DRAFTSMAN		6	51000
	5520		FILE CLERK		1	5500
	1120		CHIEF			
	1120		MECH ENGR			
	1130		ELEC ENGR			
	1330		DRAFTSMAN			
	5520		FILE CLERK			
			TOTAL-2123			
2130			COMPONENT ENGR GRP	2100		
	1120		MANAGER		1	16500
	5210		SECRETARY		1	4500
		2131	SECTION		1	43000
		2132	SECTION		1	41500
		2133	SECTION		1	49000

Figure 1-21. Hypothesis Report

HYP0 COMMENT 'INSERT QUANTITY AT VARIABLE 1,  
SALARY AT VARIABLE 2'.

EXECUTE HE.

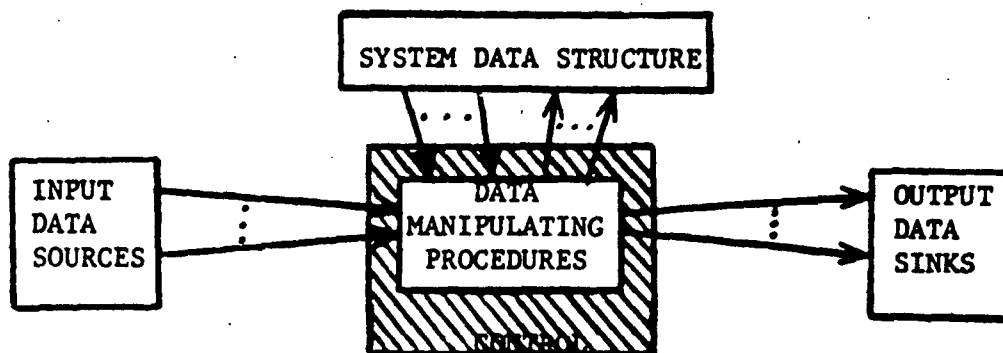
HE EXECUTE HA HB HC HD.

Figure 1-22. Hypothesis Report (Alternate Method)

## 2. COLINGO C-10

### GENERAL DESCRIPTION OF COLINGO C-10

COLINGO C-10 (hereafter called C-10) is designed to perform the usual operations of file generation, file restructuring, and data retrieval.



A schematic of data flow in the system is shown in Figure 2-1. A multiplicity of data paths interconnect the various components of the system through a set of control and manipulative procedures. C-10 has an appropriate set of languages and associated mechanisms to allow any and all of these paths to be used concurrently. This concurrency of file generation, file restructuring, and data retrieval process is one of the distinguishing characteristics of C-10.

C-10 is an essentially machine-independent data-base system realized on a small and widely used computer, the IBM 1410. The system is built upon a single, generalized data structure, a set of associated primitive data manipulators, a hierarchy of languages (with their associated translators) having differing degrees of ease of use and power, and a set of standard utility procedures.

The C-10 system imposes few restrictions on external data formats and attempts to provide a general means of structuring data within the system, as well as a general way to allow procedures to act upon data. The system is composed of a data structure, a set of machine-dependent procedures, and an "open-ended" set of system-dependent procedures.

The data structure is rich enough to enable the implementation of multi-level files, directories, dictionaries, and system-oriented data structures within one basic framework.

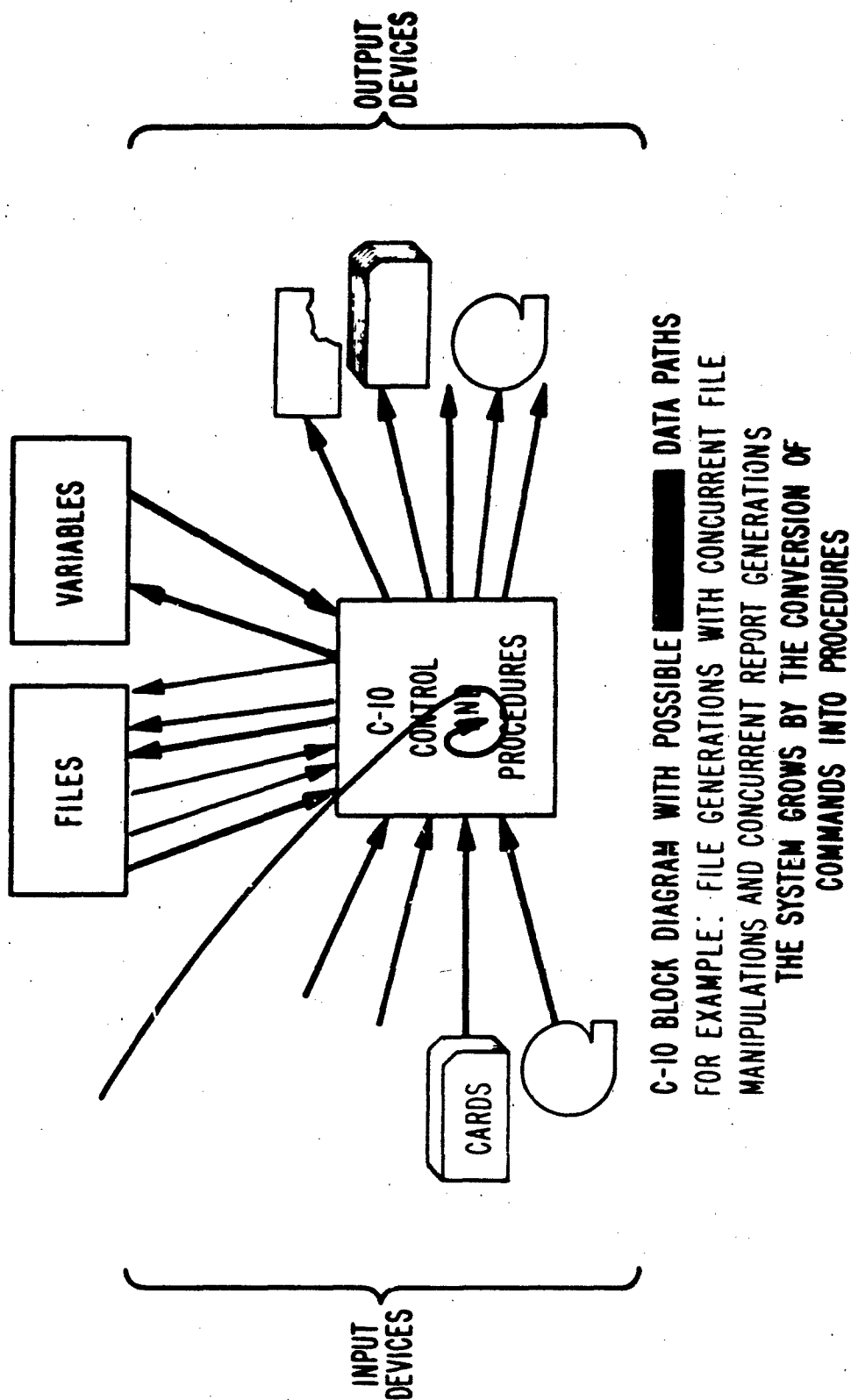


Figure 2-1. Organization of the C-10 System

Machine-independent procedures are compatible at their interfaces with those dependent procedures necessary to deal directly with the input/output hardware, the realization of the data structure, and the necessary dynamic linkage between procedures.

System-dependent procedures, the common type within C-10, may be specified in any of four hierarchical system languages (including the highest-level user language), and will be appropriately translated and executed by the system. Such procedures may remain temporary, or may be made permanent modules of the system. Examples of temporary system procedures are casual commands or queries directed to the system. "Stored queries" are examples of more permanent additions to the system made by a user.

There is a practical limit to the number of procedures which may be maintained within C-10, or restrictions on the order of appeal to these procedures, or to the total number of procedures which may be invoked at any instant; thus, any procedure may invoke any procedure at any time. The method by which these facilities are implemented results in a system which is logically independent of machine configuration (above a certain minimum), but one which is very dependent upon configuration from a performance standpoint.

The high-level system language, PROFILE, is a COBOL-like, file-structure-dependent, file-processing-oriented language. Though the language provides statements which allow for the brief and convenient specification of common bread-and-butter operations, other statements allow complete control of concurrent multiple-file manipulation operations (generation, change, testing, sorting, ...) along with control of simultaneous, concurrent input and/or output operations (limited by the number of physical devices), with provisions for handling arbitrary external data formats. ALGO-like arithmetic and Boolean statements control the typical (mixed) arithmetic processing and manipulation. PROFILE's function is to allow control of multiple-file manipulations, as well as computational, decision and input/output operations.

The file structure of C-10 allows for multiple-level tree structures of arbitrary complexity with repetitive nodes on any level. The logical realization of this file structure within the system data structure is by means of a compact representation utilizing a single marker (represented in the machine by a sine character) for each node or node repetition within a file. The realization of the system data structure may be termed serial in a block, random by block, and is oriented toward efficient use of disc storage devices. The file overhead is on the order of ten percent.

Data stored within the file structure may be in the form of arbitrary character sequences (chosen from an alphabet of 60 characters), floating-point numbers in a standard system format, or fixed-point integers. Data fields may be of fixed or variable lengths, and subfields or pseudonyms for full fields may be used in such a way that the data type differs from that of the parent field. No maximum limit is imposed upon the length of an alphabetic field.

The minimum practical machine configuration for C-10 is a small (40,000 character) central memory, a card reader/punch, a printer, and a disc. The system is capable of dynamically adjusting to operate on any normal 1410 configuration; it is capable of using any number of magnetic tape drives, additional printers, card devices, and any combination of 1301, 1302, or 1311 disc drives. Special provisions exist for driving a CRT, and one has been planned into the system. Central memories of up to 80,000 characters may be used.

A centralization of input/output control, in conjunction with central allocation of disc space, promotes high disc integrity for other users of the system.

C-10 may be driven on-line through the 1415 console (or CRT, if available), or in a batch processing mode through any input device. In the on-line mode, the system is capable of conversation with the user, and may use any of its procedures for interpretation of input (evaluation of variables, or interpretation of commands, for example).

The COLINGO-D system performs simple operations on single files efficiently, but experiences difficulty in more complex operations. The design and construction of C-10 was prompted by a desire for a system which would handle these more complex file-processing operations more naturally and efficiently.

The inclusion of primitive operations at a usable level was needed to ensure the capability of handling those unforeseen requirements which always come along. Other design goals were: ease of modification, documentation, and growth; and an attempt at greater machine-independence to gain more design carry-over for the next time.

In order to achieve the performance goal for complex operations, a penalty was accepted on simple ones. The aim of the organization was to smoothly pass multiple data streams through the system while always being prepared to bring any necessary set of procedures to bear upon the data, in contrast to COLINGO-D which repetitively passes the data through individual subsystems. The price here is in procedure-manipulation overhead. The discipline used in C-10 for implementation of such a system leads directly to the other design goals.

C-10 repetitively processes commands from the system command source; initially, this is the console typewriter, but may be changed by command to be any other input device. Commands to the system may direct it to change its mode of operation, or to perform some procedure presented to it.

Procedures (Appendix D, Page 3-82) are normally specified as a sequence of PROFILE statements. C-10 transforms these file-structure-oriented statements into equivalent system-data-structure-oriented STEP statements. STEP is a procedural language used internally within C-10.

Symbolic STEP procedures may be executed directly by an interpreter which is a part of C-10, or may be transformed by the system into directly executable machine code in the form of a C-10 subroutine.

#### SOLUTION OF THE DATA-BASE PROBLEM

The C-10 system allows considerable flexibility in the structuring of data files. One of several reasonable structures was selected for purposes of demonstrating a solution of this problem.

##### Basic Files

The selected approach uses two basic data files: one containing the organization information and the other containing the personnel information. The file sizes are approximately linear in the number of entries, while the dictionaries are fixed.

##### Organization (ORG) File

###### Structure

The ORG file structure is shown in Figure 2-2. To produce the problem reports described below, the order of the organization units in this file must be identical to the order presented in the Data Base listing (that is, by organizational hierarchy). The property TOT.PEO is computed during file generation. The file size is 3965 characters and the dictionary size is 523 characters.

###### Updating Philosophy

A change that does not affect an object size is made directly in the ORG file with appropriate changes to other properties which may be affected by the change. Any change made at the organizational, job, or unit levels causes file regeneration since it is assumed that changes at these levels will occur relatively infrequently. An example of an updating process is presented in Appendix E, Page 3-86.

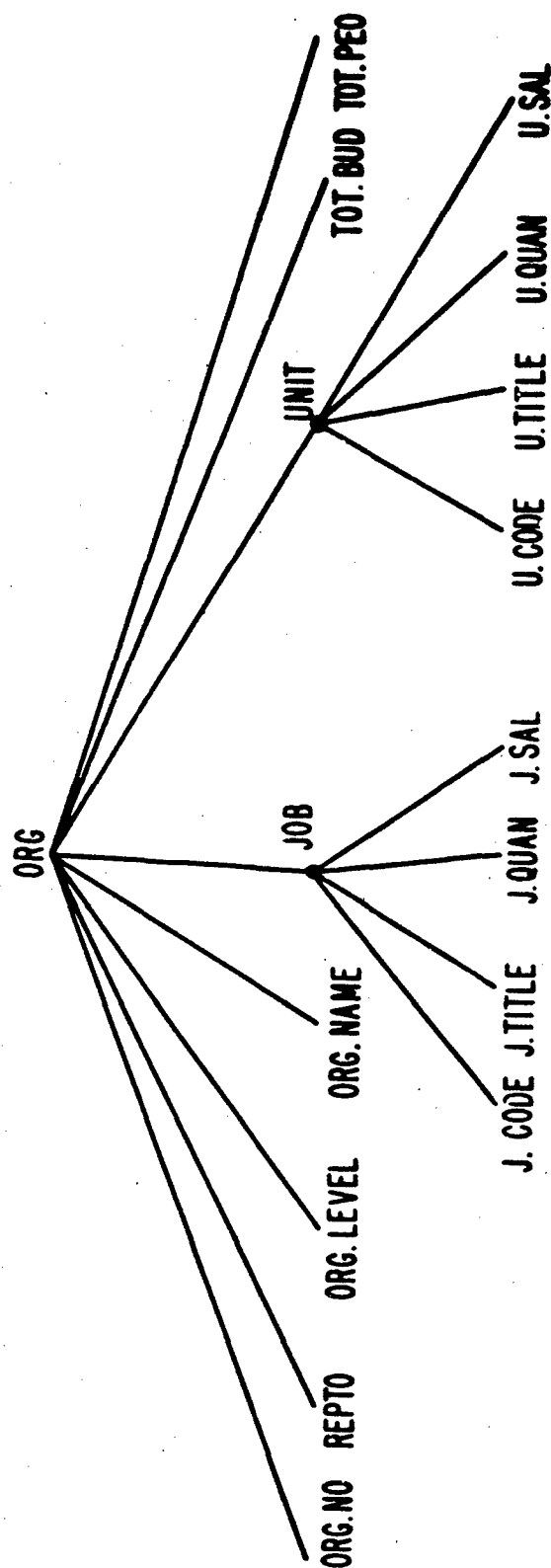


Figure 2-2. Organization File Structure  
(in C-10, ordered by Organizational Hierarchy)



Personnel (PERS) File

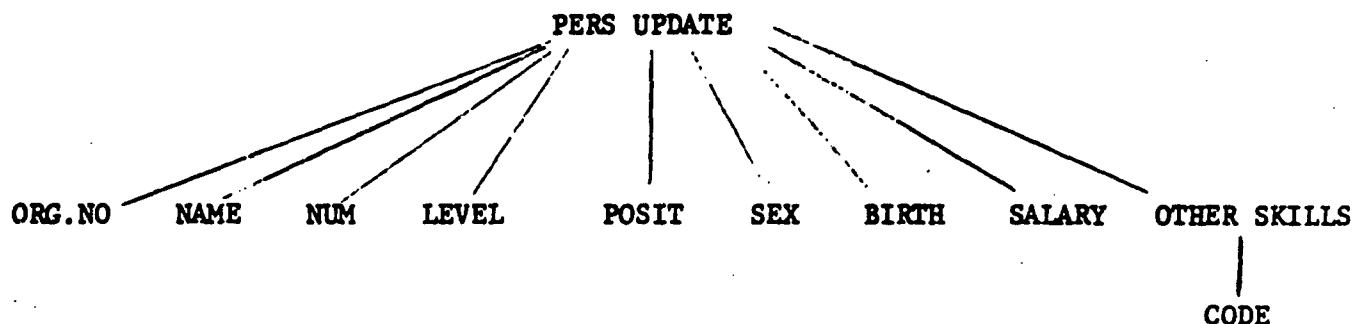
## Structure

The PERS file structure is shown in Figure 2-3 and Appendix A, Page 3-69. People are grouped under their prime job which is, in turn, a part of an organizational unit. The organizational units must be in the same order as they occur in the ORG file (i.e., by organizational hierarchy). The coding to generate the PERS file is shown in Appendix B, Page 3-71). Note that several properties (TOT.SAL, TOT.PEO, QUAN, and TOT.JOB.SAL) are computed during file generation, and that the Control Report is also produced. The three properties used in the Control Report are saved in the CONTROL file so that they may be obtained independently of PERS file generation. The file size is 11922 characters and the dictionary size is 612 characters.

## Updating Philosophy

The updating philosophy of the PERS file uses a combination of the possible methods. Changes that do not affect object size are made directly in the file (e.g., the salary change for B. E. Garber, Appendix E). Additions and deletions at the organizational and prime job levels cause file regeneration since it is assumed that changes at these levels would occur relatively infrequently.

Deletion of a person from an organization unit causes the setting of the DELETE property (e.g., DELETE would be set for S. M. Quinn). On the other hand, the addition of a person to an organization unit causes the setting of the ADD.PEO property and the generation of an object in the PER.UPDATE file:



The transfer of an individual from one organization unit to another causes the setting of the DELETE property in the old unit and the ADD.PEO property in the new unit.

Finally, three empty slots are generated in the PERS file for each OTHER.SKILL object. Thus, the new skill code, 3570, for R. E. Lee is inserted into one of his empty OTHER.SKILL slots.

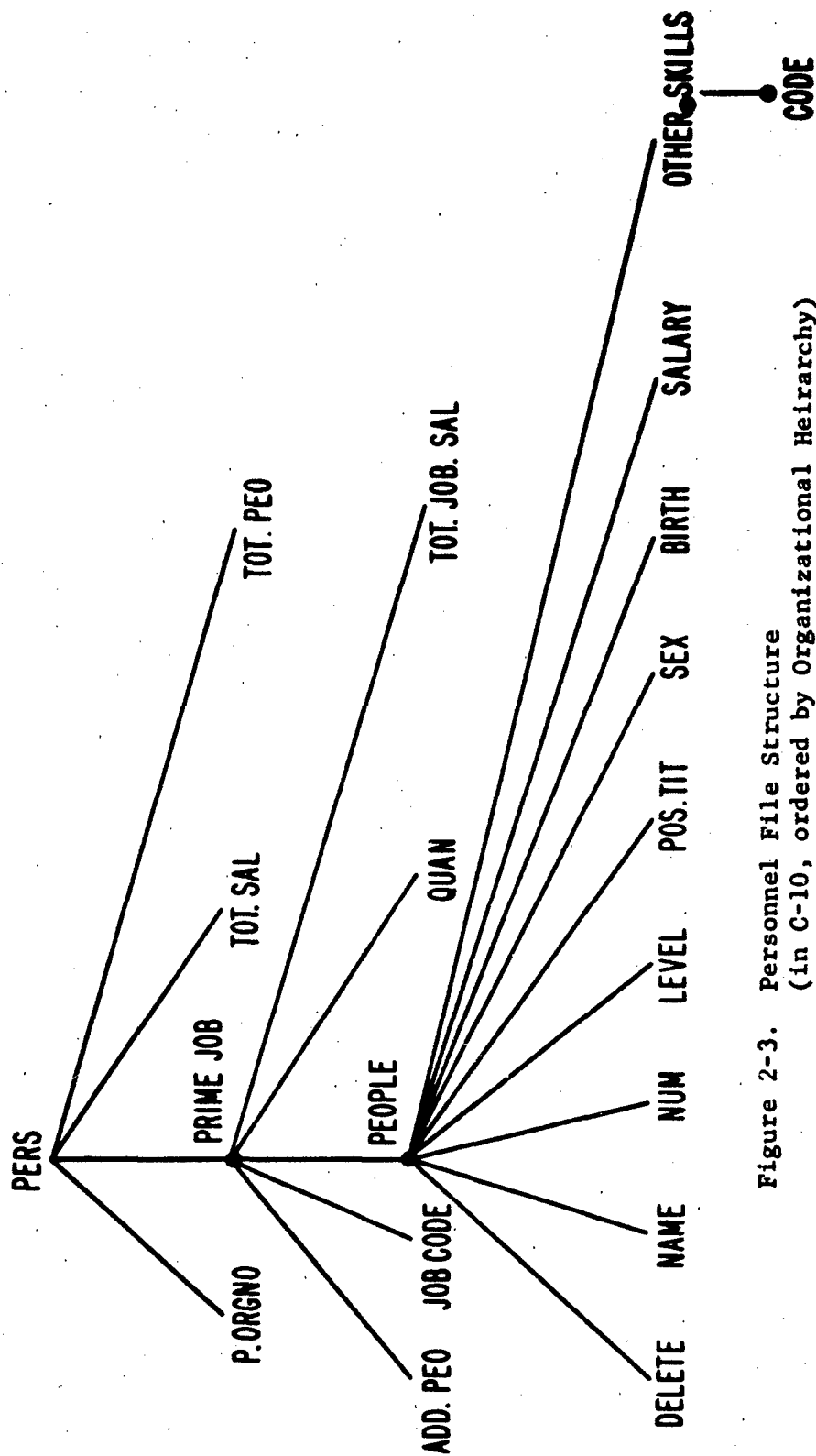


Figure 2-3. Personnel File Structure  
(in C-10, ordered by Organizational Hierarchy)

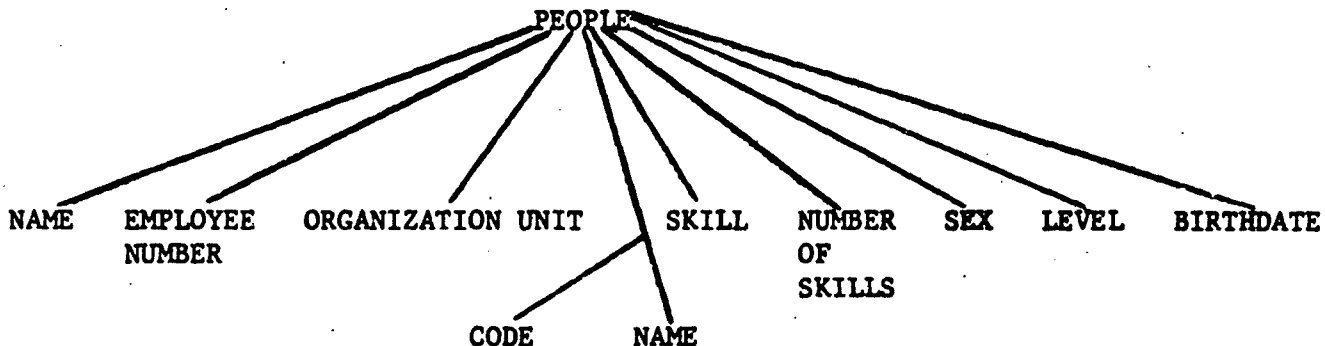
In all cases, properties computed by the file-generation program are updated approximately as file changes are made (e.g., S. M. Quinn's termination and R. E. Lee's salary change cause updates of some of the computed properties).

### Periodic and Exception Reports

The Group and Section Periodic reports and the Exception report are all produced during one concurrent pass through the ORG and PERS files. The logic and data flow is shown in Figure 2-4. The major control is at the Group level i.e., for each Periodic Group report generated, each Section that reports to that Group is processed. The Periodic Section report is generated as well as the Exception report, when relevant. Each Section also contributes data to the Group report. The Group report is printed immediately while the other two reports are saved on tapes to be printed later.

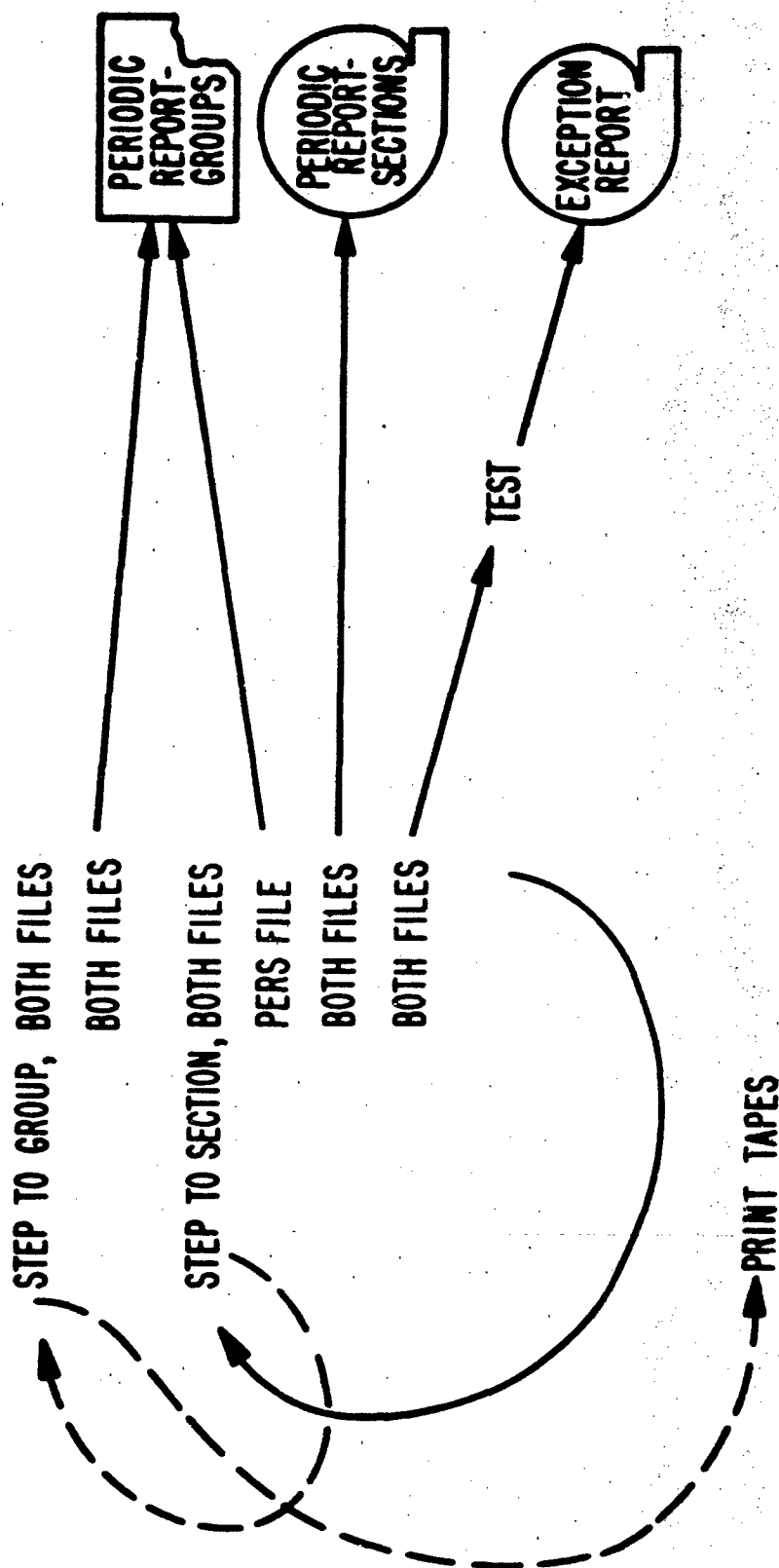
### Demand Reports

Since the PERS file is not conveniently structured to produce Demand reports, a new, semi-permanent file is made from the PERS file. The new PEOPLE file contains all of the personnel in the company and is structured by individuals rather than by organization unit, as shown below:



The coding to generate the PEOPLE file is presented in Appendix B. The constraints of the two specific Demand reports were not used to generate the PEOPLE file, since it was assumed that these Demand reports are just two of many that may be desired.

The coding to generate the two specific Demand reports is shown in Appendix C. Other Demand reports may be generated by constructing queries similar to those in Appendix C, Page 3-78. The PEOPLE1 file is used to save a subsetting file which is to be sorted and then printed again.



**LOGIC AND DATA FLOW DURING SIMPLE CROSS-FILE PROCESSING AND PREPARATION  
OF PERIODIC AND EXCEPTION REPORTS**

Figure 2-4. Periodic and Exception Reports  
(in C-10)

APPENDIX AFILE DEFINITION

REMARK - DEFINE ALL FILE STRUCTURES;  
CREATE DICTIONARY

```
PERS(  
  P.ORGNO $(I,4),  
  PRIME.JOB(  
    ADD.PEO $(A,1),  
    JOB.CODE $(I,4),  
    PEOPLE(  
      DELETE $(A,1),  
      NAME $(A,V),  
      NUM $(I,5),  
      LEVEL $(A,4),  
      POS.TIT $(A,32),  
      SEX $(A,1),  
      BIRTH $(I,6),  
      SALARY $(I,5),  
      OTHER.SKILLS(  
        CODE $(I,4)))  
      QUAN $(I,6),  
      TOT.JOB.SAL $(I,10))  
    TOT.SAL $(I,6),  
    TOT.PEO $(I,6));
```

CREATE DICTIONARY

```
CONTROL.REPORT(  
  NUMBER.OF.PERSONS.EMPLOYED $(I,6),  
  TOTAL.ACTUAL.ANNUAL.SALARIES $(I,10),  
  NUMBER.OF.ORGANIZATION.UNITS $(I,5));
```

APPENDIX A (cont)

## CREATE DICTIONARY

## PEOPLE(

NAME \$(A,V),

EMPLOYEE.NUMBER \$(I,5),

ORGANIZATION.UNIT \$(I,4),

## SKILL(

CODE \$(I,4),

NAME \$(A,V)),

NUMBER.OF.SKILLS \$(I,V),

SEX \$(A,1),

LEVEL \$(A,4),

BIRTHDATE \$(I,6)),

## CREATE DICTIONARY

## ORG(

ORG.NO \$(I,4),

REPTO \$(I,4),

ORG.LEVEL \$(A,V,7),

ORG.NAME \$(A,V,30)),

## JOB(

J.CODE \$(I,4),

J.TITLE \$(A,V,12),

J.QUAN \$(I,2),

J.SAL \$(I,6)),

## UNIT(

U.CODE \$(I,4),

U.TITLE \$(A,V,12),

U.QUAN \$(I,2),

U.SAL \$(I,6)),

TOT.BUD \$(I,6),

TOT.PEO \$(I,6)),

APPENDIX BFILE GENERATION

```

PROCEDURE GENERATE.PEOPLE.FILE ();
BEGIN;
  VARIABLE K, L;
  L1.. READ PERS; ELSE BEGIN; COMPLETE PEOPLE; RETURN; END;
  L2.. READ PRIME.JOB; ELSE GO TO L1;
  L6.. READ PERS(PEOPLE); ELSE GO TO L2;
  WRITE PEOPLE;
  NAME = NAME;
  EMPLOYEE.NUMBER = NUM;
  ORGANIZATION..NIT = P.ORGNO;
  K = 1;
  L = JOB.CODE;
  L4.. WRITE SKILL;
  CODE = L;
  L3.. READ SKILL.CODES.AND.NAMES;
      ELSE DO PRINT ('CODE',JOB.CODE,'NOT IN SKILL.CODES.AND.NAMES');
  IF L NE SKILL.CODES.AND.NAMES (CODE); GO TO L3;
  NAME = SKILL.CODES.AND.NAMES(NAME);
  CLOSE SKILL.CODES.AND.NAMES;
  READ OTHER.SKILLS; ELSE GO TO L5;
  L = OTHER.SKILLS(CODE);
  K = K+1;
  GO TO L4;
  L5.. NUMBER.OF.SKILLS = K;
  SEX = SEX;
  LEVEL = LEVEL;
  BIRTHDATE = BIRTH;
  GO TO L6;
END;

```

APPENDIX B (cont)

```

REMARK - GENERATION OF ORG FILE;
PROCEDURE GENERATE.ORG.FILE();
BEGIN;
  INPUT FROM CARDS;
  VARIABLE TEMP1, TEMP2, TOT.PEOV, TOT.BUDV;
START.. IF FIELD 5 = '*****'; BEGIN; COMPLETE ORG; RETURN; END;
        SPACE INPUT BACKWARD 4;
        WRITE.ORG;
        TOT.PEOV = 0;
        TOT.BUDV = 0;
        ORG.NO = NU(FIELD 4);
        SPACE INPUT 'DIV' OR 'DEPT' OR 'GROUP' OR 'SECTION';
        ORG.LEVEL = FIELD ' ' OR ' ';
        SPACE INPUT BACKWARD RECORD;
        SPACE INPUT 8;
        ORG.NAME = FIELD ' ';
        SPACE INPUT BACKWARD RECORD;
        SPACE INPUT 39;
        REPTO = NU(FIELD 4);
        SPACE INPUT RECORD;
        SPACE INPUT 1;
NEXT.   IF FIELD 4 NE ' ';
        BEGIN;
        SPACE INPUT BACKWARD RECORD;
        GO TO NEXT.ORG;
        END;
        SPACE INPUT 5;

```



APPENDIX B (cont)

```

TEMP1 = NU(FIELD 4);
IF TEMP1 EQ 0; GO TO UNIT;
WRITE JOB;
J.CODE = TEMP1;
SPACE INPUT 44;
TEMP1 = NU(FIELD 2);
TOT.PEOV = TOT.PEOV + TEMP1;
J.QUAN = TEMP1;
SPACE INPUT 3;
TEMP2 = NU(FIELD 6);
J.SAL = TEMP2;
TOT.BUDV = TOT.BUDV + TEMP 1 * TEMP2;
SPACE INPUT BACKWARD 43;
J.TITLE = FIELD ' ',
SPACE INPUT RECORD;
GO TO NEXT;

UNIT.. SPACE INPUT 2;
WRITE UNIT;
U.CODE = NU(FIELD 4);
SPACE INPUT 37;
TEMP1 = NU(FIELD 2);
U.QUAN = TEMP1;
SPACE INPUT 3;
TEMP2 = NU(FIELD 6);
U.SAL = TEMP2;
TOT.BUDV = TOT.BUDV + TEMP1 * TEMP2;
SPACE INPUT BACKWARD 43;
U.TITLE = FIELD ' ',
SPACE INPUT RECORD;
GO TO NEXT;

NEXT.ORG.. TOT.BUD = TOT.BUDV;
TOT.PEO = TOT.PEOV;

GO TO START;

```

APPENDIX B (cont)

```

PROCEDURE GENERATE.PERS.FILE();
BEGIN;
  VARIABLE EMPLOYEE.NUMBERV, NAMEV, UNIT.CODEV, JOB.CODEV, LEVELV,
    POSITION.TITLEV, SEXV, BIRTHDATEV, SS1V, SS2V, SS3V, SS4V,
    SALARYV, TOT.SALV, TOT.PEOV, TOT.JOB.SALV, TAAS, NOOU,
    JOB.CODEV2, UNITV, JOB.CODEVLAST, QUANV,
    UNITV = 0;
  JOB.CODEVLAST = 0;
  QUANV = 0;
  TOT.PEOV = 0;
  TOT.SALV = 0;
  TOT.JOB.SALV = 0;
  TAAS = 0;
  NOPE = 0;
  NOOU = 0;

  NEXT.SET..;
  IF SCAN.PERS.CARD.1(EMPLOYEE.NUMBERV, NAMEV, UNIT.CODEV, JOB.CODEV,
    LEVELV, POSITION.TITLEV, SEXV, BIRTHDATEV) = 0;
    GO TO END.DATA;
  DO SCAN.PERS.CARD.2(JOB.CODEV2, SS1V, SS2V, SS3V, SS4V, SALARYV);
  IF UNIT.CODEV NE UNITV,
    GO TO NEW.ORG;
  SAME.ORG..;
  IF JOB.CODEV NE JOB.CODEVLAST,
    GO TO NEW.JOB;

```

APPENDIX B (cont)

```
SAME.JOB... ,
TOT.PEOV = TOT.PEOV + 1;
QUANV = QUANV + 1;
WRITE PEOPLE;
NAME = NAMEV;
NUM = EMPLOYEE.NUMBERV;
LEVEL = LEVELV;
POS.TIT = POSITION.TITLEV;
SEX = SEXV;
BIRTH = BIRTHDATEV;
SALARY = SALARYV;
IF SS1V NE Ø;
BEGIN;
WRITE OTHER.SKILLS;
CODE = SS1V;
END;
IF SS2V NE Ø;
BEGIN;
WRITE OTHER.SKILLS;
CODE = SS2V;
END;
IF SS3V NE Ø;
BEGIN;
WRITE OTHER.SKILLS;
CODE = SS3V;
END;
IF SS4V NE Ø;
BEGIN;
WRITE OTHER.SKILLS;
CODE = SS4V;
END;
```

APPENDIX B (cont)

```
WRITE OTHER.SKILLS;
CODE = Ø;
WRITE OTHER.SKILLS;
CODE = Ø;
WRITE OTHER.SKILLS;
CODE = Ø;
TOT.SALV = TOT.SALV + SALARYV;
TOT.JOB.SALV = TOT.JOB.SALV + SALARYV;
GO TO NEXT.SET;

NEW.JOB..;
IF JOB.CODEV NE Ø;
BEGIN;
QUAN = QUANV;
TOT.JOB.SAL = TOT.JOB.SALV;
END;
WRITE PRIME.JOB;
JOB.CODE = JOB.CODEV;
JOB.CODEVLAST = JOB.CODEV;
QUANV = Ø;
TOT.JOB.SALV = Ø;
GO TO SAME.JOB;

NEW.ORG..;
IF UNITV NE Ø;
BEGIN;
QUAN = QUANV;
TOT.JOB.SAL = TOT.JOB.SALV;
TOT.SAL = TOT.SALV;
TOT.PEO = TOT.PEOV;
END;
```

APPENDIX B (cont)

```
WRITE PERS,  
  P.ORGNO = UNIT.CODEV,  
  UNITV = UNIT.CODEV,  
  JOB.CODEV = β,  
  TOT.SALV = β,  
  QUANV = β,  
  TOT.PEOV = β,  
  TOT.JOB.SALV = β,  
  TAAS = TAAS + TOT.SALV,  
  NOPE = NOPE + TOT.PEOP,  
  NOOU = NOOU + 1,  
  GO TO SAME.ORG;  
  
END.DATA..;  
TOT.SAL = TOT.SALV;  
TOT.JOB.SAL = TOT.JOB.SALV;  
TOT.PEO = TOT.PEOV;  
QUAN = QUANV;  
COMPLETE PERS;  
  
REMARK - GENERATE CONTROL REPORT FILE;  
WRITE CONTROL.REPORT,  
  NUMBER.OF.PERSONS.EMPLOYED = NOPE+TOT.PEOV,  
  TOTAL.ACTUAL.ANNUAL.SALARIES = TAAS+TOT.SALV,  
  NUMBER.OF.ORGANIZATION.UNITS = NOOU,  
  COMPLETE CONTROL.REPORT,  
  RETURN;  
END;
```

APPENDIX CREPORT

```

REMARK: GENERATE TWO DEMAND REPORTS;
DO GENERATE.PEOPLE.FILE ( ),
PRINT AND WRITE SUBSET (PEOPLE1) PEOPLE
IF ANY SKILL(CODE) EQ 334$
AND NUMBER.OF.SKILLS GT 2
AND SEX EQ 'M'
AND LEVEL NE 'HEAD'
AND EXTRACT (BIRTHDATE, 4, 2) GT 16 AND LT 35
(NAME, EMPLOYEE.NUMBER, ORGANIZATION.UNIT, SKILL(CODE, NAME)),
SORT PEOPLE1 ON PEOPLE1(NAME),
PRINT SUBSET PEOPLE1
(NAME, EMPLOYEE.NUMBER, ORGANIZATION.UNIT, SKILL(CODE, NAME));
DELETE FILE PEOPLE1;
PRINT AND WRITE SUBSET (PEOPLE1) PEOPLE
IF SEX EQ 'M'
AND LEVEL NE 'HEAD'
AND ANY [SKILL(CODE)EQ 111$ OR SKILL (CODE) GT 1129 AND LT 114$]
(NAME, EMPLOYEE.NUMBER, ORGANIZATION.UNIT, SKILL(CODE, NAME));
SORT PEOPLE1 ON PEOPLE1(NAME),
PRINT SUBSET PEOPLE1
(NAME, EMPLOYEE.NUMBER, ORGANIZATION.UNIT, SKILL(CODE, NAME));
DELETE FILE PEOPLE1;
DELETE FILE PEOPLE;
RETURN;

```

APPENDIX C (cont)

```

PROCEDURE GENERATE.REPORTS ( );
BEGIN;
  VARIABLE GROUP.TOTAL.1, GROUP.TOTAL.2, SECTION.TOTAL.1,
    SECTION.TOTAL.11, SECTION.TOTAL.2, SECTION.TOTAL.22,
    EX.REPORT, LAST.GROUP;
  DEFINE BLOCK FOR OUTPUT = 132; REMARK - SET FOR 1403 PRINTER;
  REMARK - THIS PROCEDURE GENERATES--
    1. THE PERIODIC SECTION REPORT,
    2. THE PERIODIC GROUP REPORT,
    3. THE EXCEPTION REPORT
    ONLY ONE PASS IS MADE OVER THE FILES;
  FIRST.. READ ORG; ELSE COMMENT 'ORG FILE WITHOUT GROUPS OR SECTIONS';
    IF ORG.LEVEL EQ 'DIV' OR 'DEPT'; GO TO FIRST;
    IF ORG.LEVEL NE 'GROUP'; COMMENT 'ORG FILE NOT ORDERED PROPERLY';
    OUTPUT TO TAPE 'TAPE1';
    LAST.GROUP = 0;
  START.. DO WRITE.Heading (ORG.NO, REPTO, ORG.NAME);
    GROUP.TOTAL.1 = 0; GROUP.TOTAL.2 = 0;
  JOBS.. READ JOB; ELSE GO TO END.JOBS;
    DO WRITE.LINE (0, J.CODE, J.TITLE, J.QUAN, J.SAL);
    GROUP.TOTAL.1 = GROUP.TOTAL.1 + J.SAL;
    GO TO JOBS;
  END.JOBS.. READ UNIT; ELSE GO TO END.UNITS;
    DO WRITE.LINE (1, U.CODE, U.TITLE, U.QUAN, U.SAL);
    GROUP.TOTAL.1 = GROUP.TOTAL.1 + U.SAL;
    GO TO END.JOBS;

```

APPENDIX C (cont)

```

END.UNITS.. DO WRITE.TOTALS(' ', GROUP.TOTAL.1);
DO WRITE.AC();
MATCH.. READ PERS; ELSE COMMENT 'ORG UNIT NOT REPRESENTED IN PERS FILE';
IF ORG.NO NE P.ORGNO; GO TO MATCH;
NEXTP.. READ PRIME.JOB; ELSE GO TO END.PRIME.JOBS;
DO WRITE.LINE(0, JOB.CODE, TITLE.LOOKUP(JOB.CODE), QUAN, TOT.JOB.SAL);
GROUP.TOTAL.2 = GROUP.TOTAL.2 + TOT.JOB.SAL;
GO TO NEXTP;
END.PRIME.JOBS.. READ ORG; ELSE
BEGIN;
LAST.GROUP = 1;
GO TO END.GROUP;
END;
IF ORG.LEVEL EQ 'DIV' OR 'DEPT'; GO TO END.PRIME.JOBS;
IF ORG.LEVEL EQ 'GROUP'; GO TO END.GROUP;
OUTPUT TO PRINTER;
EX.REPORT = 0;
BACK.. DO WRITE.Heading(ORG.NO, REPTO, ORG.NAME);
SECTION.TOTAL.1 = 0; SECTION.TOTAL.11 = 0;
SECTION.TOTAL.2 = 0; SECTION.TOTAL.22 = 0;
JOBS2.. READ JOB; ELSE GO TO END.JOBS2;
DO WRITE.LINE(0, J.CODE, J.TITLE, J.QUAN, J.SAL);
SECTION.TOTAL.1 = SECTION.TOTAL.1 + J.SAL;
SECTION.TOTAL.11 = SECTION.TOTAL.11 + J.QUAN;
GO TO JOBS2;
END.JOBS2.. DO WRITE.TOTALS(SECTION.TOTAL.11, SECTION.TOTAL.1);
DO WRITE.AC();
MATCH2.. IF EX.REPORT = 1; GO TO NEXTP2;
READ PERS; ELSE COMMENT 'ORG UNIT NOT REPRESENTED IN PERS FILE';
IF ORG.NO NE P.ORGNO; GO TO MATCH2;

```



APPENDIX C (cont)

```

NEXTP2.. READ PRIME.JOB; ELSE GO TO END.PRIME.JOB2;
DO WRITE.LINE (0, JOB.CODE, TITLE.LOOKUP(JOB.CODE), QUAN, TOT.JOB.SAL);
SECTION.TOTAL.2 = SECTION.TOTAL.2 + TOT.JOB.SAL;
SECTION.TOTAL.22 = SECTION.TOTAL.22 + QUAN;
GO TO NEXTP2;

END.PRIME.JOB2.. DO WRITE.TOTALS(SECTION.TOTAL.22, SECTION.TOTAL.2);
DO WRITE.DEVIATION(SECTION.TOTAL.2 - SECTION.TOTAL.1);
IF EX.REPORT = 1; GO TO GSEC;
IF SECTION.TOTAL.22 GE SECTION.TOTAL.11 AND
SECTION.TOTAL.2 GE SECTION.TOTAL.1 + 800;
BEGIN; REMARK - MUST DO EXCEPTION REPORT;
EX.REPORT = 1;
OUTPUT TO TAPE 'TAPE2';
GO TO BACK;
END;

GSEC.. OUTPUT TO TAPE 'TAPE1';
DO WRITE.LINE (1, ORG.NO, 'SECTION', SECTION.TOTAL.22,
SECTION.TOTAL.2);
GROUP.TOTAL.2 = GROUP.TOTAL.2 + SECTION.TOTAL.2;
GO TO END.PRIME.JOB2;

END.GROUP.. OUTPUT TO TAPE 'TAPE1';
DO WRITE.TOTALS(0, GROUP.TOTAL.2);
DO WRITE.DEVIATION (GROUP.TOTAL.2 - GROUP.TOTAL.1);
IF LAST.GROUP NE 1; GO TO START;
DO PRINT.TAPE ('TAPE1');
DO PRINT.TAPE ('TAPE2');
RETURN;
END;

```

1 December 1965

3-82

TM-2624/100/00

APPENDIX D

PROCEDURES

```
PROCEDURE SCAN.PERS.CARD.1 (,EMPLOYEE.NUMBERD,NAMED,UNIT.CODED,  
  JOB.CODED,LEVELD,POSITION.TITLED,SEXD,BIRTHDATED);  
  BEGIN;  
    VARIABLE TEMP;  
    INPUT FROM CARDS;  
    IF FIELD 5 EQ '*****',  
      BEGIN;  
        SCAN.PERS.CARD.1 = 0;  
        RETURN;  
      END;  
    SPACE INPUT BACKWARD 4;  
    EMPLOYEE.NUMBERD = NU(FIELD 5);  
    NAMED = FIELD 19;  
    UNIT.CODED = NU(FIELD 4);  
    SPACE INPUT 1;  
    JOB.CODED = NU(FIELD 4);  
    IF FIELD 4 = 'HEAD';  
      BEGIN;  
        TEMP = INPINTER();  
        POSITION.TITLED = FIELD ',' OR ' ';  
        SPACE INPINTER()+32-TEMP;  
      END;  
    ELSE;  
      POSITION.TITLED = FIELD 32;  
      SEXD = FIELD 1;  
      BIRTHDATED = NU(FIELD 6);  
      SCAN.PERS.CARD.1 = 1;  
      SPACE INPUT 3;  
    RETURN;  
  END;
```

APPENDIX D (cont)

```
PROCEDURE SCAN.PERS.CARD.2 (,JOB.CODED,SS1,SS2,SS3,SS4,SALARYD);  
  BEGIN;  
    INPUT FROM CARDS;  
    SPACE INPUT 1;  
    JOB.CODED = NU(FIELD 4);  
    SPACE INPUT 1;  
    SS1 = NU(FIELD 4);  
    SPACE INPUT 1;  
    SS2 = NU(FIELD 4);  
    SPACE INPUT 1;  
    SS3 = NU(FIELD 4);  
    SPACE INPUT 1;  
    SS4 = NU(FIELD 4);  
    SPACE INPUT 1;  
    SALARYD = NU(FIELD 4);  
    SPACE INPUT 50;  
    RETURN;  
  END;
```

```
PROCEDURE WRITE.HEADING(ORG.UNIT, REPORTS.TO, ORG.NAME);  
  BEGIN;  
    FIELD = 'ORG.UNIT';  
    SPACE OUTPUT 5;  
    FIELD = ORG.UNIT;  
    SPACE OUTPUT BLOCK;  
    FIELD = 'REPORTS TO';  
    SPACE OUTPUT 3;  
    FIELD = REPORTS.TO;  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT BLOCK;  
    FIELD = 'ORG NAME';  
    SPACE OUTPUT 5;  
    FIELD = ORG.NAME;  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT BLOCK;  
    FIELD = 'AUTHORIZED COMPLEMENT';  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT BLOCK;  
    RETURN;  
  END;
```

1 December 1965

3-84

TM-2624/100/00

APPENDIX D (cont)

```
PROCEDURE WRITE.LINE(KEY, CODE, TITLE, QUAN, SAL);  
  BEGIN;  
    SPACE OUTPUT 5+KEY*5;  
    FIELD 4 = CODE;  
    SPACE OUTPUT 10-KEY*5;  
    FIELD 12 = TITLE;  
    SPACE OUTPUT 2;  
    FIELD 2 = QUAN;  
    SPACE OUTPUT 8;  
    FIELD 6 = SAL;  
    SPACE OUTPUT BLOCK;  
    RETURN;  
  END;
```

```
PROCEDURE WRITE.TOTALS(Q.TOTAL, S.TOTAL);  
  BEGIN;  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT 19;  
    FIELD = 'TOTAL';  
    SPACE OUTPUT 9;  
    FIELD 1 = Q.TOTAL;  
    SPACE OUTPUT 18;  
    FIELD 6 = S.TOTAL;  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT BLOCK;  
    RETURN;  
  END;
```

1 December 1965

3-85

TM-2624/100/00

APPENDIX D (cont)

```
PROCEDURE WRITE.AC();  
  BEGIN;  
    FIELD = 'ACTUAL COMPLEMENT';  
    SPACE OUTPUT BLOCK;  
    SPACE OUTPUT BLOCK;  
    RETURN;  
  END;
```

```
PROCEDURE WRITE.DEVIATION(DEVIAION);  
  BEGIN;  
    SPACE OUTPUT 19;  
    FIELD = 'DEVIATION FROM BUDGET';  
    SPACE OUTPUT 13;  
    FIELD 6 = DEVIATION;  
    SPACE OUTPUT BLOCK;  
    RETURN;  
  END;
```

1 December 1965

3-86

TM-2624/100/00

APPENDIX E

UPDATES

REMARK - UPDATING TRANSACTIONS;

PROCESS PERS;

IF NUM = 33144 AND NAME = 'QUINN,S.M.'; CHANGE DELETE TO 1;

IF NUM = 91152 AND NAME = 'GARBER,B.E.'; CHANGE SALARY TO 8500;

IF NUM = 85657 AND NAME = 'LEE,R.E.';

BEGIN;

IF NOT ANY CODE = 0; DO ENTER.PERS.UPDATE (PEOPLE, 3570);

IF CODE = 0; CHANGE CODE TO 35700;

END;

### 3. ADAM

#### GENERAL DESCRIPTION

ADAM is a computer program system being built for the MITRE-ESD Systems Design Laboratory as a tool for use in the design and evaluation of Military Information Systems. ADAM operates on the IBM 7030 Computer under control of a monitor system associated with the computer facility.

ADAM is used in the early stages of system design to simulate the external appearance of systems which include man-machine communication and relatively large data bases. In addition, the development and use of ADAM is expected to provide a vehicle through which techniques of data-management system design and programming can be compared and evaluated. Thus, ADAM is a developmental system as opposed to a production or operational system--it is intended to do a laboratory job, in a laboratory which studies alternative system designs and system techniques.

The two major concepts behind ADAM are:

- (1) Generalize system functions most likely to be required in a real-time, multi-user, on-line data handling environment: file generation, file maintenance, query processing, report generation, etc.
- (2) Provide a method for specifying operations on data independent of the form or format of the data (e.g., field length, number of items).

The design of ADAM incorporates features which accomplish these aims while trying to maintain an acceptable level of operating efficiency.

#### Data Structures

The major data structures of ADAM are the file structure and the roll structure. Files store all data. They are variable in length and flexible in format. Rolls are a combined dictionary directory for the files.

#### File Structure

A file is organized as a series of entries, each of which has an identical structure; i.e., all entries in one file are characterized by the same properties. Data is stored as property values, usually one value for each property in each entry. An entry may contain substructures called repeating groups which are collections of associated properties with repeated sets of values. A repeating group has all the structure of a file; one repetition of each of its property values corresponds in structure to one entry in a file. Repeating groups may, in turn, contain repeating groups.

Property types presently available in ADAM files are:

- Numeric - Numeric data is stored as an integer in a fixed-size field big enough to contain its declared largest value or as a floating-point number in a one-computer-word field.
- Logical - Data which represents names is stored in a fixed-length field as a code with code name equivalences kept in a roll.
- Raw - Variable-length, non-numeric data is stored in a variable-length field.
- Repeating Group A repeating-group property is actually an associated set of properties and their values. Each repetition contains a value for each property in the group. All values for properties in a group are stored within the file entry in which they belong.

Property-value storage is variable to the extent that:

- (a) A raw-property value takes up only as much space as it needs, which may be different from one value of the property to another.
- (b) A repeating group takes up space only for repetitions which exist. The number of repetitions may vary from one entry or parent repeating group to another.
- (c) As a consequence of (a) and (b) above, entries in the same file may differ in length.
- (d) The variability in entry size is dynamic. Entries may be changed during ADAM operation without rewriting or copying the file.

#### File Accessing

Each file has one property (a logical property) whose values name entries. Data may be retrieved randomly by direct access to a specific entry through the dictionary (roll) which contains the entry name. This roll also contains the current location of the entry. Alternatively, all entries in a file may be serially accessed. Within an entry, access to a specific repetition of a repeating group is always serial; the entry is selected and each repetition of the group examined until the required one is located.

Files may contain as data (in the form of logical-valued properties) the names of other files, entries, or properties, thus allowing cross-file and cross-entry linkages.



## Rolls

A roll is a combined dictionary/directory. As a dictionary, it stores names, which may be multi-word and variable-length, along with a fixed-length code used as internal representation. Names which are values of logical-valued properties are stored in rolls; e.g., MALE and FEMALE as values for the property SEX would be stored in a roll (the file would contain the corresponding fixed-length code number). In addition, rolls contain the names of entities in ADAM--file names, property names, entry names, etc. One function of rolls as dictionaries is to allow the compressed internal code representation, a second is to allow synonyms; i.e., M and MALE, if declared synonymous, would have the same internal code.

As a directory, a roll stores the dynamically changing information which describes the current physical location and the current format of ADAM entities files, properties, objects, input-output terminals, etc.

## Roll Accessing

Rolls are usually accessed implicitly. During input-message translation, the translator program looks up input names and transforms them to internal codes; before output, the output formatting program transforms outputs back into name form.

## Language and Format Description

The ADAM translator has been designed to translate a class of languages, rather than a particular language. When the translator is called upon to translate a message, it retrieves from a file a description of the particular language to be used. A description of a language consists of a set of recursive procedures written in a special interpretive code; each recursive procedure corresponds to a syntactic unit of the language. Language descriptions are stored in a language file. The output of the translator consists of a set of interpretive instructions which are subsequently executed by a processor to carry out the action specified by the message.

The handling of formats for outputs follows a design principle similar to that for languages. User-specified format descriptions reside in a format file. Messages or programs which specify output also select one of the available format descriptions in the file. An output formatting program interpretively executes the format specifications.

In the initial version of the ADAM system, the language file contains a description of a basic file-generation language and a basic retrieval language. The format file contains a selection of standard formats for display, typewriter, and printer output. For any specific application, desired language(s) and format(s) may be described and their descriptions inserted into the files.

### String Substitution - Ad Hoc Language Definition

In addition to the ability to specify descriptions for complete languages, the user of ADAM may define single words as "string-substitutions." These words are replaced by their definitions before an input message is translated. String substitution definitions may specify that parameters (comprising the words following the defined word) be inserted. Thus, the message

LET SUM MEAN (CHANGE /3/ TO /1/ + /3/).

defines a substitution. The message segment

...SUM A IN B ...

would be transformed to

...CHANGE B TO A+B ...

before translation.

Substitutions are defined as being effective for all words from specified I/O devices, in which respect they differ from synonyms (for file, object, and property names) which are effective for references to the specific file.

### Subroutines

The ADAM compiler, called DAMSEL, accepts intermixed assembly language, macro language, and ADAM-oriented statements. Compiled programs must be inserted into the ADAM routine file to be used. In that sense, subroutines are considered as "data" for many ADAM operations. Subroutines in the routine file may be executed by other subroutines or as a result of message inputs.

DAMSEL itself allows the usual complement of arithmetic-assignment, conditional, and subroutine-call statements, and includes a macro facility for extending the DAMSEL language. In addition, it provides statements specifically designed to create, augment, modify, and retrieve from ADAM data structures. File-manipulation statements refer directly to files by name, or use names which the subroutine receives as input parameters from other routines or from a message input. In either case, statements in the subroutine are independent of the format of data referred to; data descriptions are retrieved from the rolls when a subroutine is compiled or executed.

Although the DAMSEL compiler constitutes the major means for writing user-specific subroutines, other compilers may be used. In particular, an available ADAM system routine adjusts the interface of subroutines compiled by FORTRAN to be compatible, so that computational routines may be coded in FORTRAN for convenience.

### Message Processing

ADAM message processing is shown in Figures 3-1 and 3-2. Messages originate from real-time devices (typewriters, display light-pencils, etc.) or off-line as simulated real-time inputs. As each message is received, the system is interrupted to determine its priority, to determine the language in which it is written, and to stack the message. This is the input cycle. When the execution cycle (of internal processing routines) completes a task, it requests another message, and the message with the highest priority is unstacked.

Major programs in the execution cycle are the Translator, Processor and Output Generator. After string substitution (not shown), the Translator, with reference to the appropriate language specification from the Language File, translates the message into a list of things to be done, called a process table. The Processor executes the steps specified in the process table interpretively, accessing or modifying the data base and operating any routines specified. If output is produced, it is always in the form of a file. Any output formatting required is performed on the data from this file, as described by a format from the format file.

The output formatting program delivers the output to be stacked, pending availability of an output device. The execution cycle then requests another message from the input cycle. Meanwhile, the output cycle delivers material to devices as fast as the equipment permits and accepts output for stacking at any time. Thus, the three cycles (input, execute, output) operate asynchronously from one another.

### THE DATA-BASE PROBLEM

A system to solve the data-base problem could be designed in many ways which might differ in convenience, ease of implementation, or efficiency. ADAM would be applied to the design problem--alternative designs would be modeled in ADAM and the resulting systems exercised to optimize design parameters. The solution presented here is one such model.

In constructing a model in ADAM, a user must describe his problem in four areas: data structures, languages, subroutines, and formats. For this problem, the general decisions in these areas are:

- |                     |  |
|---------------------|--|
| (1) Data Structures | Design files to match input data closely.  |
| (2) Languages       | Do not create a new language specification; use an existing language modified with string substitutions. |

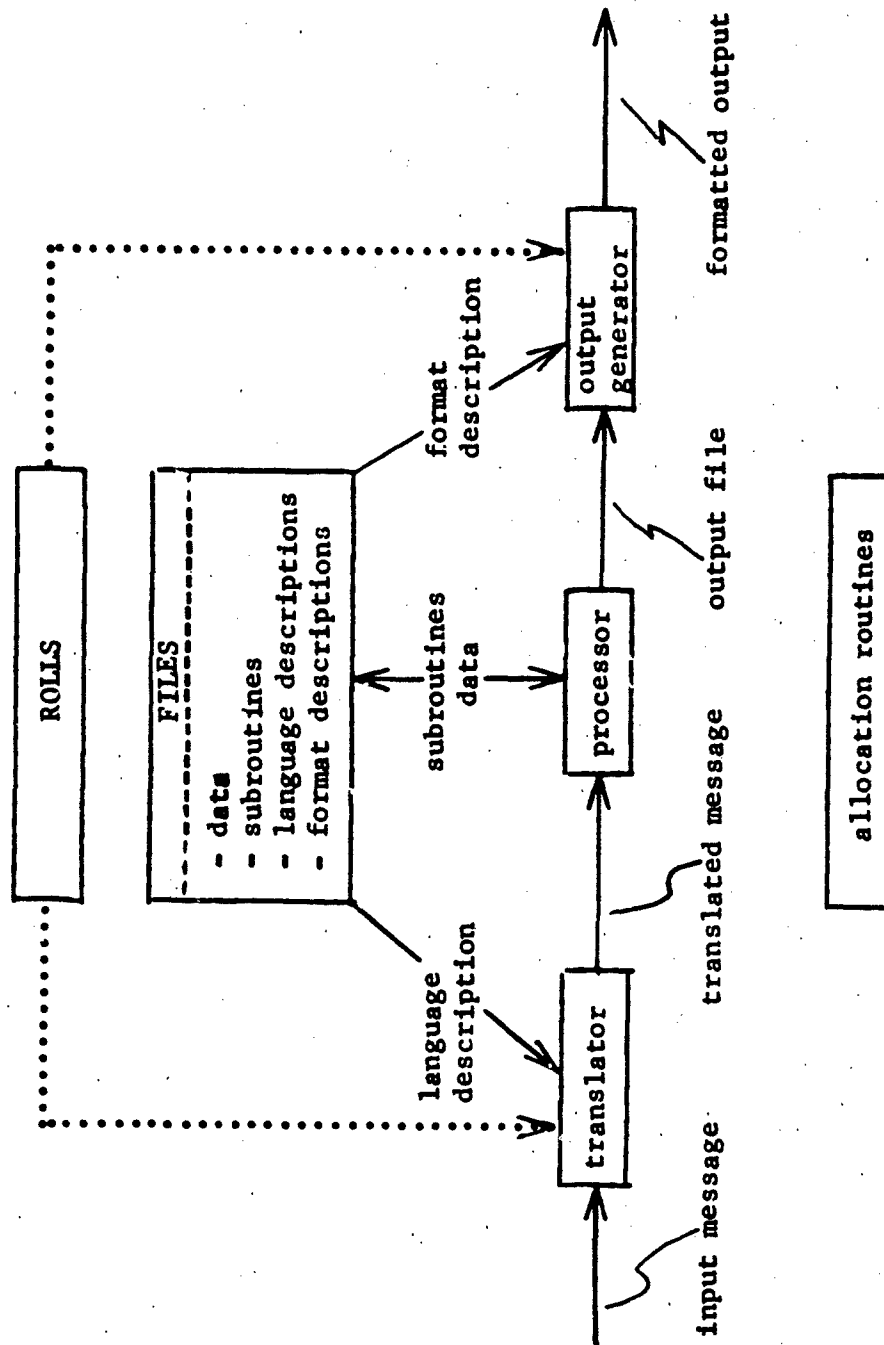


Figure 3-1. Basic Execution Cycle

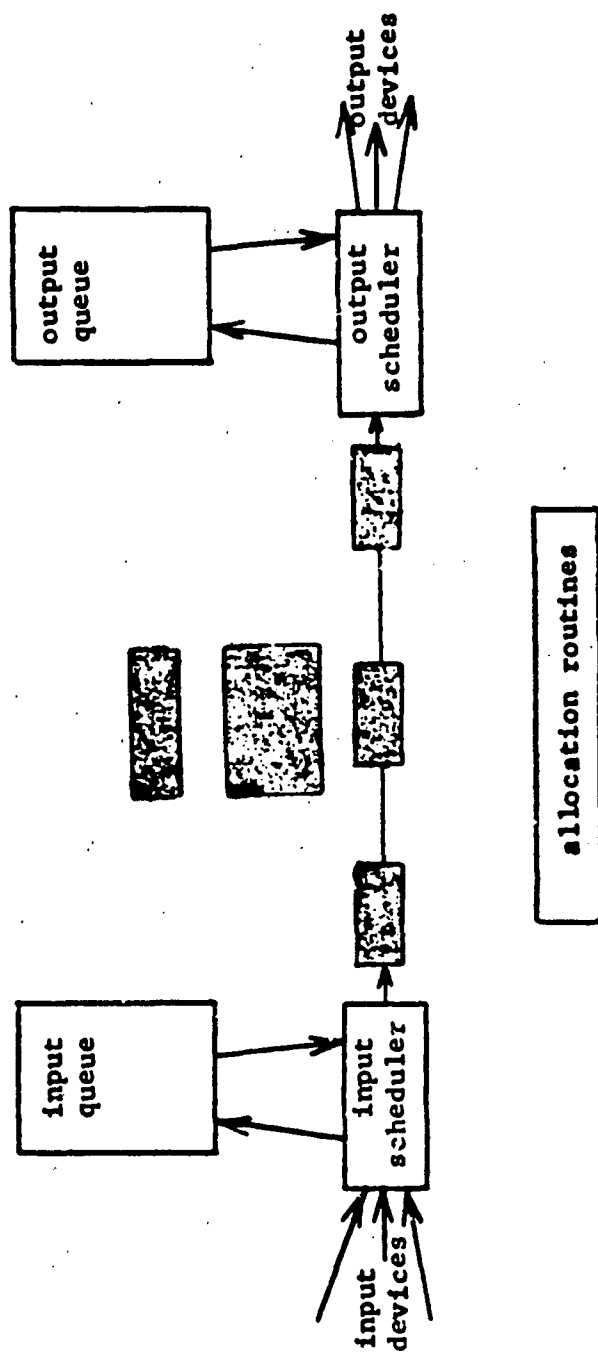


Figure 3-2. Input-Executive-Output

- (3) Routines      Do not write routines for data-base processing and report generation; use the message-language exclusively. As an example, write one routine to do table lookups on skill codes.
- (4) Formats      Devise one special format for periodic, exception, and hypothesis-testing reports; otherwise use existing formats.

These decisions were made to simplify the design of a first model. In actual practice, the first model would be evaluated and modified designs made on the basis of experience with the model. Alternatives might be:

- (a) Redesign file-structure for greater convenience or efficiency
- (b) Design a special-purpose language tailored to this data base
- (c) Perform some or all operations with subroutines instead of messages
- (d) Design different report formats as required.

In the design of this solution, the following assumptions about the problem were made:

- (1) Updating of the Personnel File is the most frequent transaction, and is normally done by specifying the employee number.
- (2) Demand reports, of the kind in the example, are frequently requested and require rapid response.
- (3) Periodic reports occur less frequently than updating transactions and demand reports. The actual complement is computed only when the periodic report is requested. For this reason, changes in personnel are not reflected into the Organization File when the Personnel File is updated. The Exception Report is assumed to be generated only after a periodic report is requested and hence uses the output of the periodic report.
- (4) Hypothesis Testing Reports are less frequently requested than updating and demand reports.
- (5) Although the organization data is shown with units sorted in hierarchical order, the solution never assumes that it will be generated that way or be kept in order in the file.

#### Data Structures

The two types of records, organization and personnel, are kept in two separate files. In addition, as files are a convenient structured storage and can be created and deleted dynamically, temporary files are created during message processing to store intermediate results. In ADAM, all output reports are in the form of files. One output file (the ORGSECT file) is saved after being printed for use as data. Exclusive of output files, the following files participate:

1 December 1965

3-95

TM-2624/100/00

<u>NAME</u>	<u>USE</u>	<u>ENTRY</u>	<u>PROPERTY WHICH NAMES ENTRY</u>	<u>CREATED BY</u>
ORG	primary data	organizational unit	UNIT	file-generation
PERS	primary data	person	EMPL NO	file-generation
CONTROL	control report	one-entry only	no name needed	file-generation
ORGSECT	section report, data input for group and exception reports	organizational unit (same structure as ORG file)	UNIT	section-report message
TEMP	intermediate data for hypothesis report	organizational unit (same structure as ORG file with only certain sections represented)	UNIT	hypothesis-report message

along with the following rolls (created during file generation):

<u>ROLL NAME</u>	<u>NAMES IN ROLL</u>	<u>VALUES IN ROLL</u>
ORG PROPERTY ROLL	ORG file properties	file formats for properties
ORG OBJECT ROLL	ORG file entries	location of entries
PERS PROPERTY ROLL	PERS file properties	file formats for properties
PERS OBJECT ROLL	PERS file entries	location of entries
L	levels (e.g., HEAD, EMPL...)	-----
X	sexes	-----
TYPE	unit type (e.g., DIV, DEPT..)	-----

and the property and object rolls for the intermediate files created during message processing, and deleted thereafter. If desired, the information in the TYPE, L, and X rolls could be kept in a single roll, at the cost of some validity checking (MALE would then be a legal value for LEVEL and EMPL a legal value for SEX, and both would be legal values for TYPE).

## Properties

Property names, types, and field sizes are defined in the file-generation messages which make the files. In addition, synonyms for property names are defined to shorten them in retrieval messages; e.g., AUTHORIZED COMPLEMENT (AUTH). Files created as a result of retrieval messages take their property names and descriptions from already defined files.

In general, properties whose values are numbers are declared "numeric." Exception to this practice must be made for properties which identify entries in files (UNIT in the organization file, EMPL NO in the personnel file) which are required to be logical properties.

For cross-reference purposes, all other properties which refer to units are similarly declared logical-type, with values defined in the same roll as that for entry names in the organization file. Properties which are non-numeric are declared logical-properties, (e.g., LEVEL, TYPE) with the exception of NAME and TITLE, which are declared raw-properties to illustrate handling of variable-length file data. NAME and TITLE could equally as well be logical-properties stored in encoded form in fixed-length fields in the file.

The use of repeating groups is fairly straightforward. Defined as a repeating group containing MONTH, DAY, and YEAR, the property BIRTHDATE illustrates the use of that type of property in a non-repeated situation, where it serves as a collective name for three values, any of which may be accessed individually by name. A request to retrieve BIRTHDATE will yield all three.

## Organization File

The organization file treats sections, groups, department, and divisions identically, each as an entry identified by the property UNIT which contains as its value the unit number. The hierarchical structure of the company is described by the property REPORTS TO which contains as its value the unit number of the parent organizational unit and by the property AUTH UNIT CODE which contains unit numbers of subunits. File entries in the organization file may be accessed by reference to these properties. In addition to properties which correspond to information on the data cards, new properties (e.g., AUTHORIZED TOTAL QUANTITY, and ACTUAL TOTAL BUDGET) are defined as places to accumulate material for reports; the property values are empty when the file is generated. Finally, the property TYPE (= section, group, department, division) was added to make retrieval simpler.

The structure of the organization file is shown in Figure 3-3.



1 December 1965

3-97

TM-2624/100/00

properties

one sample entry

UNIT

2000

TYPE

DIVISION

NAME

REPRESENTATIVE DIV

REPORTS TO

1000

AUTHORIZED

COMPLEMENT

JOB CODE

0110

5210

----

----

UNIT CODE

----

----

2100

2100

TITLE

DIVISION MANAGER

SECRETARY

DEPT

DEPT

QUANTITY

1

1

1

1

SALARY

28000

6000

510900

425800

TOTAL AUTHORIZED

970700

BUDGET

TOTAL AUTHORIZED

QUANTITY

—

ACTUAL

COMPLEMENT

JOB CODE

—

UNIT CODE

—

TITLE

—

QUANTITY

—

SALARY

—

TOTAL ACTUAL

BUDGET

—

TOTAL ACTUAL

QUANTITY

—

Figure 3-3. Organization File

## Personnel File

The personnel file has one entry for each person, identified by the property EMPL NO. It contains the information from the data cards. The property UNIT in a personnel file entry contains as values the names of entries in the organization file and can be used to access directly the unit to which a person is assigned. A new property, TOTAL SKILLS is added as a place to count skills. The structure of the personnel file is shown in Figure 3-4.

## Input Data

Appendix D, Page 3-121, shows the code sheets for the input data for file generation. The format duplicates that of the input data given with the problem with the following exceptions:

- (a) A terminator character, either \* or -, is added at the end of each variable-length field, and at the end of each repetition of each repeating group.
- (b) Values for the property TYPE of organization have been added.
- (c) The personnel data, too long for a single card, is arbitrarily split into two cards, the second card beginning just after birthdate data. On the second card, salary is moved to the beginning of the card to allow for additional skill codes.

## Language

The messages used in this problem are expressed in one of two existing languages: file generations are expressed in a file-description language, while report and updating transactions use a retrieval language called FABLE. A short summary of the rules of FABLE is given in Appendix B, Page 3-117. String substitutions define abbreviations for long names and for more complex operations in either language.

The use of FABLE as a query and update language prevents the use of the full capabilities of ADAM; i.e., even if ADAM can do a job, it might not be possible to say what to do in FABLE, or at least easily to say what to do.

Alternatives available are to specify a different language, or to write routines instead of messages to accomplish retrieval and updating.

1 December 1965

3-99

TM-2624/100/00

		entries →	
↓		properties	
<u>EMPL NO</u>		52467	28347
<u>NAME</u>		MILLER	KOPLEY
<u>UNIT</u>		2311	2311
<u>JOB CODE</u>		3340	3340
<u>LEVEL</u>		EMPL	EMPL
<u>TITLE</u>		MACHINE OPR	MACHINE OPR
<u>SEX</u>		M	M
<u>BIRTHDATE</u>			
	<u>MONTH</u>	08	06
	<u>DAY</u>	08	20
	<u>YEAR</u>	20	36
<u>PRIMARY SKILL</u>		3340	3340
<u>SECONDARY</u>	<u>SKILLS</u>	3510   7320   3570	3355
<u>SALARY</u>		9000	6000
<u>TOTAL SKILLS</u>		—	—

Figure 3-4. Personnel File

### Routines

One special routine is included here as an example of subroutine use. It performs a table lookup on skill names. Further development would probably store skill names in a roll with automatic lookup, but it would be necessary to discover the implied sequence of skill codes (what does the phrase "skill code >1129 and <1140" really mean?) before skills could be treated as names.

The routine is called automatically by the output generator program whenever the (numeric) skills properties are printed. It is written in the DAMSEL compiler language, compiled, and added to the routine file. (As a consequence of this treatment of skills, the numeric value is never output, a potentially undesirable situation.)

### Formats

A single format, named F1, is included to serve periodic reports, the exception report, and the hypothesis testing report, whose outputs are all essentially the same. All other outputs, by not specifying any format, use a standard format which tabulates (one entry at a time) property names in a single column with values immediately to the right of the corresponding names.

The reports which use format F1 are distinguished from one another by their titles. Some properties (e.g., DEVIATION) are missing from some reports; these are not specified as output in the queries which produce the reports and therefore are not formatted. For the hypothesis testing report, the new group "PROPOSED COMPLEMENT" is output with values from ACTUAL COMPLEMENT. The ACTUAL COMPLEMENT properties are used to store the values as they are calculated and the name changed upon output. Format F1 is called in the examples via the string substitution REPORT. (See Appendix A, Page 3-113.)

The formats of outputs are exactly the suggested formats with the addition of a title to each report.

### Message Processing and "Runs"

The concept of 'run' has little meaning to ADAM. FABLE statements may be combined into messages--a complete message is translated before it is processed. Messages may be input on-line from typewriters, teletypes or other on-line devices, or off-line as cards. In either case, message words may be separated by any number of blanks. Off-line message inputs are punched on cards according to the same spacing conventions. Card boundaries are not significant: messages may appear on several cards.

During a time period when ADAM is running on the IBM 7030 computer, many messages may be entered, on-line or off-line. Alternatively, each FABLE statement could be entered separately, its results saved and examined while ADAM was not on the computer, then restored for the next "run." For the purposes of this problem, we use the word "pass" to describe the processing associated with a single FABLE statement.

#### Data Flow for Retrieval and Maintenance

Each FABLE message can cause several files to be accessed, several or all entries in a file to be changed, and an output file to be produced. The sequence of such processing for the various operations (each individual FABLE message made up of several statements) is shown in Table 3-1. Files are accessed serially when the names of their entries are not available (e.g., persons whose level = head). Files are accessed randomly if the names of the entries to be processed can be stated explicitly in the message (e.g., PERS 33144) or the names of the entries to be processed appear in the data base (e.g., in the authorized complement of some other entry or a property such as Unit or Reports To of some other file).

Batching of requests such as the two demand reports is possible in ADAM, but the FABLE language does not permit subsetting of a file into two files.

Batching could most easily be accomplished by defining a property in the queried file to be a tag, using a FABLE message to retrieve the data for all reports in the batch in a single output file, and either sorting the output file on the tag property or querying it on a specified value of the tag.

1 December 1965

3-102

TM-2624/100/00

OPERATION	FILES ACCESSED SERIALLY	FILES ACCESSED RANDOMLY AND ACCESS KEY	DATA CHANGED	OUTPUT
Control Report	PERS ORG	----- -----	CONTROL file CONTROL file	----- Control Report
Updating	----	PERS by Empl. No.	PERS file	-----
Section Report	ORG PERS ORG	----- ORG by Unit in PERS file -----	"actual" data zeroed "actual" data updated ----	----- ----- Section Report and ORGSECT file
Group Report	ORGSECT ORG	ORG by "Reports to" in ORGSECT file -----	"actual" data updated -----	----- Group Report
Demand Report 1	PERS	-----	-----	Demand Report
Demand Report 2	PERS	-----	-----	Demand Report
Exception Report	ORGSECT	-----	-----	Exception Report
Hypothesis Testing Report	----	ORG by "authorized complement" of units in "authorized comple- ment" of ORG 2100 ORG 2123	"actual" data for sections revised "actual" data for Section 2123 revised	TEMP file contains record of changes -----
	TEMP			
	----	ORG by "authorized complement" of ORG 2100	"actual" data for groups revised	-----
	----	ORG by Unit	"actual" data for Unit 2100 revised	-----
	----	ORG by Unit	----	Hypothesis Report

Table 3-1. Data Flow Sequence

## A SOLUTION OF THE DATA BASE PROBLEM

The operations involved in the solution would proceed as follows:

- (a) Compile the SKILCODE routine and insert it in the ADAM routine file (Appendix D, page 3-121.
- (b) Compile format F1 and insert it in the ADAM format file (Appendix C, page 3-120.
- (c) Operating the various string substitution messages which define abbreviations to be used (Appendix A, page 3-113.
- (d) Operate the file generation messages for the ORG, PERS and CONTROL files.
- (e) When desired, and as frequently as desired, operate messages to generate:
  - (1) The control report
  - (2) The file updates
  - (3) The periodic reports
  - (4) The demand reports or other reports like them
  - (5) The exception report
  - (6) The hypothesis-testing report
- (f) As experience dictates, revise and repeat steps (a) through (d) above to optimize the system design.

The messages to perform steps (a) through (e) above are listed here. They have not been run with ADAM.

### File Generation

A file description message begins with the file name and source of data. Then, for each property, its type, name, and synonyms are given, followed by positioning information to locate the data. Finally, an input-data conversion routine and specific information dependent on property type complete the description. The conversion routines specified here are ALPHA and DECIMAL (see Appendix A), which convert characters and decimal numbers from card-code to internal alphanumeric and binary respectively. The length of variable-field input fields is specified by a terminator field, one or more characters long.

The CONTROL file illustrates an empty file generation.

The PERS file illustrates the handling of variable fields in the positioning statements:

SCAN TO '-', in which the character is assumed to be added to the input to terminate variable-length name fields,

SCAN TO '\*' or ',' in which the \* is assumed to be added to the input to terminate variable-length titles and the ',' is there already as a terminator, and

1 December 1965

3-104

TM-2426/100/00

TERMINATED BY '\*' in which the '\*' is assumed to be added to the input at the end of every birthdate to mark the end of the repeating group.

The generation of the PERS file, in addition to implicitly creating rolls which describe its objects and properties, explicitly creates the rolls L to contain level names and X to contain the values for the property SEX. For the properties PRIMARY SKILL and secondary SKILLS, the CONVERT OUT phrase is used to specify the SKILCODE routine as the conversion for all output of these property values.

The ORG file generation makes extensive use of synonyms to shorten long property names.

In all the file-generation messages, spaces have been inserted to make columns for improved readability. The messages could be run with or without the words spaced out this way.

MESSAGE TO GENERATE CONTROL FILE

GENERATE FILE, CONTROL, NULL. BEGIN OBJECT.

INTEGER,	NO PERSONS,	NULL DATA.	6 DIGITS
INTEGER,	TOTAL ACTUAL SALARIES,	NULL DATA.	10 DIGITS
INTEGER,	NO UNITS,	NULL DATA.	END OBJECT.

MESSAGE TO GENERATE PERSONNEL FILE

GENERATE FILE, PERSONNEL(PERS), CARDS, BEGIN OBJECT.

LOGICAL,	OBJECT NAME (EMPL NO),	LENGTH 5 COL.	ALPHA.USE OBJECT ROLL.
RAW,	NAME,	SPACE 1 COL. LENGTH VARIABLE, SCAN TO '- '.	ALPHA.PRINT STANDARD.
LOGICAL,	UNIT,	SPACE TO NON' ' . LENGTH 4 COL.	ALPHA.USE OBJECT ROLL OF ORG FILE.
INTEGER,	JOB CODE,	SPACE 1 COL. LENGTH 4 COL.	DECIMAL.4 DIGITS.
LOGICAL,	LEVEL,	SPACE 1 COL. LENGTH 4 COL.	ALPHA.USE NEW ROLL L.



1 December 1965

3-105

TM-2624/100/00

MESSAGE TO GENERATE PERSONNEL FILE (cont)

RAW,	TITLE,	SPACE 1 COL. LENGTH VARIABLE. SCAN TO '*' or ','. ALPHA.PRINT STANDARD.	
LOGICAL,	SEX,	RESET TO OBJECT START. SPACE 69 COL. LENGTH 1 COL.	ALPHA.USE NEW ROLL X .
BEGIN GROUP,	BIRTHDATE,	TERMINATED BY '*'. SPACE 1 COL.	BEGIN REPETITION.
INTEGER,	MONTH,	LENGTH 2 COL.	DECIMAL.2 DIGITS.
INTEGER,	DAY,	LENGTH 2 COL.	DECIMAL.2 DIGITS.
INTEGER,	YEAR,	LENGTH 2 COL.	DECIMAL.2 DIGITS. END REPETITION.
END GROUP,	BIRTHDATE.		
INTEGER,	SALARY,	SPACE TO NEXT CARD.	DECIMAL.6 DIGITS.
INTEGER,	PRIMARY SKILL,	SPACE 1 COL. LENGTH 4 COL.	DECIMAL.CONVERT OUT SKILNAME.4 DIGITS.
BEGIN GROUP,	SECONDARY,	TERMINATED BY '*'. SPACE 1 COL.	BEGIN REPETITION.
INTEGER,	SKILL,	LENGTH 4 COL.	DECIMAL.CONVERT OUT SKILNAME.4 DIGITS. END REPETITION
END GROUP.	SECONDARY,		
INTEGER.	TOTAL SKILLS,		NULL DATA.2 DIGITS.
END OBJECT.		SPACE TO NEXT CARD.	

1 December 1965

3-106

TM-2624/100/CO

MESSAGE TO GENERATE ORGANIZATION FILE

GENERATE FILE,ORG,CARDS.BEGIN OBJECT.

LOGICAL,	OBJECT NAME (UNIT),	LENGTH 4 COL.	DECIMAL.USE OBJECT ROLL.
LOGICAL,	TYPE,	LENGTH 3 COL.	ALPHA.USE NEW ROLL TYPE.
RAW,	NAME,	SPACE 1 COL. LENGTH VARIABLE. SCAN TO '*'.  RESET TO OBJECT START. SPACE 35 COL. LENGTH 4 COL.	ALPHA.PRINT STANDARD.   DECIMAL.USE OBJECT ROLL.
BEGIN GROUP,	AUTHORIZED COMPLIMENT (AUTH),	TERMINATED BY '*'.  JOB CODE,	BEGIN REPETITION.
INTEGER,	JOB CODE,	SPACE TO NEXT CARD LENGTH 4 COL.	DECIMAL.4 DIGITS.
LOGICAL,	UNIT CODE,	SPACE 1 COL. LENGTH 4 COL.	ALPHA.USE OBJECT ROLL.
RAW,	TITLE,	SPACE 1 COL. LENGTH VARIABLE. SCAN TO '*'.  QUANT,	ALPHA.PRINT STANDARD.  DECIMAL.2 DIGITS
INTEGER,	QUANT,	SPACE TO NON ' ' . LENGTH 2 COL.	
INTEGER,	SALARY,	SPACE 1 COL. LENGTH 6 COL.	DECIMAL.6 DIGITS. END REPETITION.
END GROUP,	AUTHORIZED COMPLEMENT.		
INTEGER,	TOTAL AUTHORIZED QUANTITY (QAUTH),		NULL DATA.4 DIGITS.
INTEGER,	TOTAL AUTHORIZED BUDGET (BAUTH),	SPACE 1 COL. LENGTH 7 COL.	DECIMAL.7 DIGITS.

MESSAGE TO GENERATE ORGANIZATION FILE (cont)

BEGIN GROUP,	ACTUAL COMPLEMENT (ACTUAL),	TERMINATED BY '*'. SPACE TO NEXT CARD. LENGTH 4 COL.	BEGIN REPETITION.  DECIMAL.4 DIGITS.
INTEGER,	JOB CODE,	SPACE 1 COL. LENGTH 4 COL.	ALPHA.USE OBJECT ROLL.
LOGICAL,	UNIT CODE,	SPACE 1 COL. LENGTH VARIABLE. SCAN TO '*'. SPACE TO NON ' '.	ALPHA.PRINT STANDARD.
RAW,	TITLE,	SPACE 1 COL. LENGTH 2 COL.	DECIMAL.2 DIGITS.
INTEGER,	QUANT,	SPACE 1 COL. LENGTH 6 COL.	DECIMAL.6 DIGITS. END REPETITION.
INTEGER,	SALARY,	SPACE 1 COL. LENGTH 6 COL.	DECIMAL.6 DIGITS. END REPETITION.
END GROUP	ACTUAL COMPLEMENT.		
INTEGER,	TOTAL ACTUAL QUANTITY (QACTUAL),		NULL DATA.4 DIGITS.
INTEGER,	TOTAL ACTUAL BUDGET (BACTUAL),		NULL DATA.7 DIGITS.
END OBJECT.		SPACE TO NEXT CARD.	

Control Report

The control report is generated by the following message:

FOR CONTROL.

REPEAT (FOR PERS. SUM (PERS SALARY) IN (TOTAL ACTUAL SALARIES), STEP  
(NO PERSONS))

AND REPEAT (FOR ORG. STEP (NO UNITS)),  
PRINT ALL.

which operates as follows:

FOR CONTROL	accesses CONTROL file, empty to start
REPEAT (FOR PERS...	iterates through all entries of PERS file, summing values of SALARY in CONTROL and counting once for each entry
REPEAT (FOR ORG...	iterates through all entries of ORG file, counting once for each entry
PRINT ALL	prints the three properties of the single entry in the CONTROL FILE

The output would appear exactly as the problem requires.

Commonly used subexpressions have been abbreviated by the terms SUM and STEP, defined in Appendix A.

#### Updating Transactions

Messages to effect updating transactions are:

FOR PERS 33144. DELETE OBJECT.

FOR PERS 91152. CHANGE SALARY TO 8500.

FOR PERS 85657. ADD REPETITION SECONDARY (SKILL=3570) AND STEP (TOTAL SKILLS).

Each message accesses the single entry specified.

#### Periodic Report

The procedure for producing the periodic reports is presented below; each paragraph corresponds to a FABLE statement as shown in Figure 3-5.

(1) For each ORG entry, erase old "actual" data and sum authorized quantities for all sections.

(2) For each person listed in the PERS file, get his unit in the ORG file. If his job type already appears, add his salary to actual salary and step actual quantity by 1. If not, add a new job type to actual complement with his job code, his salary, and his title and set actual quantity to 1. In any event, step actual total quantity and add his salary to actual total budget.

(3) For each section in the ORG file, print the required data. Name the output file ORGSECT and save it for future use.

- (1) FOR ORG. CLEAR AND IF TYPE EQ SECTION  
REPEAT (FOR AUTH. SUM AUTH QUANT IN QAUTH).
- (2) FOR PERS ALTER ORGUNIT.  
IF (ANY ORGUNIT ACTUAL JOB CODE EQ PERS JOB CODE  
STEP (ORGUNIT ACTUAL QUANT),  
SUM SALARY IN (ORGUNIT ACTUAL SALARY)  
OR ELSE  
ADD REPETITION ACTUAL (JOB CODE = JOB CODE  
SALARY = SALARY  
QUANT = 1  
TITLE = TITLE))  
AND STEP (ORGUNIT QACTUAL),  
SUM SALARY IN (ORGUNIT BACTUAL).
- (3) FOR ORG. IF TYPE EQ SECTION,  
REPORT 'GROUP REPORT'  
PRINTLIST, 'DEVIATION FROM BUDGET' = BAUTH - BACTUAL.  
NAME ORGLIST.  
SORT ON UNIT.
- (4) FOR ORGSECT ALTER ORG (REPORTS TO).  
ADD REPETITION ACTUAL (UNIT CODE = UNIT CODE  
QUANT = 1  
TITLE = SECTION  
SALARY = SALARY),  
SUM BACTUAL IN (ORG(REPORTS TO) BACTUAL).
- (5) FOR ORG. IF TYPE EQ GROUP, REPORT 'GROUP REPORT'  
PRINTLIST, 'DEVIATION FROM BUDGET' = BAUTH - BACTUAL.  
SORT ON UNIT.

Figure 3-5 Messages to Produce Periodic Reports

(4) For each entry in the ORGSECT file get the entry in the ORG file it reports to. Add a new unit to its actual complement, with actual data equal to that from the ORGSECT entry, and increase its budget by the total actual budget of the ORGSECT entry.

(5) For each group in the ORGFILE, print the required data.

Commonly used subexpressions perform the functions presented below; each is defined by a string substitution message as listed in Appendix A.

CLEAR            For the current entry in the ORG file change to 0 the actual budget and quantity and erase any 'actual' data.

ORGUNIT          That entry in the ORG file whose name is contained in the property UNIT in the PERS file.

PRINTLIST       The properties common to the section and group reports.

REPORT           Print according to format F1 with the indicated title.

#### Demand Report

The two Demand Reports are produced separately by the queries shown below. Skill names are produced automatically by the SKILCODE Routine. The word SKILCODE is an abbreviation for PRIMARY SKILL, SECONDARY SKILLS (see Appendix A).

#### MESSAGES TO PRODUCE DEMAND REPORTS

FOR PERS	IF	SEX EQ M AND LEVEL NOT EQ HEAD AND BIRTHDATE YEAR GR 16 AND LS 35 AND ANY SKILCODE EQ 3340 AND TOTAL SKILLS GR 2,
	PRINT	NAME,NUMBER,UNIT,SKILCODE,SORT ON NAME.
FOR PERS	IF	SEX EQ M AND LEVEL NOT EQ HEAD AND ANY(SKILCODE EQ 1100 OR SKILCODE GR 1129 AND LS 1140
	PRINT	NAME,NUMBER,UNIT,SKILCODE.SORT ON NAME.

#### Exception Report

The Exception Report queries the file which produced the section report and extracts a subset of its entries.

MESSAGE TO PRODUCE EXCEPTION REPORT

FOR OBJECT I7 QACTUAL GR QAUTH AND ABSOLUTE DEVIATION GQ 800, REPORT  
'EXCEPTION REPORT' ALL.

Hypothesis Testing Report

The procedure for producing the hypothesis testing report is as follows:  
each paragraph corresponds to a FABLE statement as shown in Figure 3-6.

- (1) For all section in department 2100, clear their actual complements and recompute proposed actual complements based on authorized complement minus salaries and quantities for job-code 1330. Save, in a file called TEMP, the quantity and salary for all jobs with job code 1330 (i.e., those not added-in in the recomputation). Entries in this file will correspond to sections from which draftsmen were withdrawn.
- (2) For section 2123, add a repetition to actual complement for job code 1330, with empty quantity and salary, then add from the TEMP file the salaries and quantities previously saved.
- (3) For the units which report to department 2100, clear the actual complements and recompute proposed actual complements.
- (4) For Department 2100, clear the actual complements and recompute proposed actual complements.
- (5) Print the resulting output as if it were a periodic report, but select the sections and groups to print based on the given sequence.

The messages use the following string substitutions, given in Appendix A.

REPORT	Print according to format F1 with the indicated title.
SUBUNIT	The set of units in the authorized complement of this department.
CLEAR	Erase actual data.
RECOMPUTE	The phrases to compute an authorized complement excluding instances of a given job-code.
LEVEL0, LEVEL1, LEVEL2	Specification of unit level for RECOMPUTE.
ORGLIST	The selection of units in the ORG file to be printed.
PRINTLIST1	The set of "authorized" data to be printed.
SUBSUBUNIT	The set of units in the authorized complements of SUBUNITS.

- (1) FOR ORG 2100 ALTER SUBSUBUNIT.  
 CLEAR AND RECOMPUTE LEVEL2 JOB 1330 AND  
 IF SUBSUBUNIT ACTUAL JOB CODE EQ 1330,  
 SAVE 'QUANT' = ACTUAL QUANT,  
       'SALARY' = ACTUAL SALARY.  
 NAME TEMP.
- (2) FOR ORG 2123.  
 ADD REPETITION ACTUAL (JOB CODE = 1330,  
       TITLE = DRAFTSMAN,  
       SALARY = 0,  
       QUANT = 0 ) AND  
 IF ANY (ACTUAL JOB CODE EQ 1330  
       REPEAT (FOR TEMP.CHANGE ACTUAL QUANT TO ACTUAL QUANT + TEMP QUANT,  
               ACTUAL SALARY TO ACTUAL SALARY + TEMP SALARY,  
               QACTUAL TO QACTUAL + TEMP QUANT,  
               BACTUAL TO BACTUAL + TEMP SALARY)).
- (3) FOR ORG 2100 ALTER SUBUNIT. CLEAR AND RECOMPUTE LEVEL1 JOB 1330.
- (4) FOR ORG 2100. CLEAR AND RECOMPUTE LEVEL0 JOB 1330.
- (5) FOR ORGLIST. REPORT 'PROPOSED TRANSFER.DRAFTSMEN TO SECTION 2123'  
 PRINTLIST1, 'PROPOSED COMPLEMENT' = ACTUAL,QACTUAL,BACTUAL.

Figure 3-6. Messages for Hypothesis Report



APPENDIX A - STRING SUBSTITUTIONS

All string substitution messages used in the solution are listed in this Appendix.

ALPHA, DECIMAL

LET ALPHA MEAN (CONVERT USING CAA).  
LET DECIMAL MEAN (CONVERT USING CDB).

The two abbreviations above simplify the conversion phrase in file-generation messages. CAA and CDB are the names of conversion routines.

SUM, STEP, COUNT

LET SUM MEAN (CHANGE /3/ TO /3/ + /1/).  
LET STEP MEAN (CHANGE /1/ TO /1/ + 1).  
LET COUNT MEAN (REPEAT (FOR 1/1.STEP/3/))

SUM adds two values together. It is used in phrases of the form SUM A IN B. The "noise" word IN is eliminated because the second parameter, /2/, is not mentioned in the definition.

STEP adds the number 1 to a property value

COUNT is an analog of STEP which incorporates the FABLE construction "REPEAT (...)" to iterate through multiple occurrences of entries in a file or repetitions of a repeating group.

CLEAR

LET CLEAR MEAN (CHANGE BACTUAL TO Ø, QACTUAL TO Ø AND DELETE GROUP ACTUAL)

CLEAR removes old "actual" values and their totals from the ORG file entry being accessed. For example:

FOR ORG.CLEAR.

would delete actual values from every entry of the ORG file.

REPORT

LET REPORT MEAN (PRINT FORMAT F1 TITLE)

REPORT abbreviates output phrases by including the specification for format F1 and the word TITLE. It is used in Periodic and Exception reports.

PRINTLIST, PRINTLIST1

LET PRINTLIST1 MEAN (NAME, REPORTS TO, AUTH, QAUTH, BAUTH).  
LET PRINTLIST MEAN (PRINTLIST1, ACTUAL, QACTUAL, BACTUAL).

PRINTLIST1 is the collection of properties in the ORG file common to the Section, Group, and Hypothesis Report outputs. It excludes actual data properties which are printed as "proposed" in the Hypothesis Report.

PRINTLIST includes the actual data properties and is used in the Periodic and Exception Reports.

ORGLIST

LET ORGLIST MEAN (ORG 2111, 2113, 2115, 2110, 2122, 2123, 2120, 2131,  
2132, 2133, 2135, 2130, 2190, 2100)

ORGLIST is the ordered list of entries in the ORG file to be printed in the Hypothesis Report.

ORGUNIT

LET ORGUNIT MEAN (ORG (UNIT))

ORGUNIT is that entry on the ORG file whose name is contained in the property UNIT in the PERS file. For example:

FOR PERS 2113 ALTER ORGUNIT.PRINT NAME.

would print the name of the entry in the ORG file which was the unit to which person 2113 reported.

SKILLCODE

LET SKILLCODE MEAN (PRIMARY SKILL, SECONDARY SKILL).

SKILLCODE is an abbreviation for the properties PRIMARY SKILL and SECONDARY SKILL taken together. It is used in the Demand Report messages.

RECOMPUTE

LET RECOMPUTE MEAN(

REPEAT (IF FOR /1/. /1/AUTH JOB CODE NOT EQ/4/ AND GR 0 RESET /1/ OR IF /1/  
AUTH JOB CODE NOT GR 0

ADD REPETITION ACTUAL (UNIT CODE = /1/AUTH UNIT CODE,  
TITLE = /1/AUTH TITLE,  
QUANT = /1/AUTH QUANT,  
SALARY = /2/ BACTUAL))

AND CHANGE BACTUAL TO /1/ BACTUAL + /1/ ACTUAL SALARY,  
QACTUAL TO /1/ QACTUAL + /1/ ACTUAL QUANT, OR ELSE))

RECOMPUTE stands for a sequence of FABLE phrases which specify operations on the units indicated by its first parameter (one of LEVEL0, LEVEL1, LEVEL2 defined above) to generate a proposed complement consisting of the authorized complement minus the instances of the job code specified as the second parameter.

For example, RECOMPUTE LEVEL1 JOB 5210 will set the proposed complement of all units directly attached to the department equal to the authorized complement, excluding instances of job code 5210.

RECOMPUTE also specifies operations to calculate Total Proposed Quantity and Total Proposed Budget.

RECOMPUTE proceeds as follows:

REPEAT((IF FOR /1/...	Start a pass through the specified entries
...AUTH JOB CODE NOT EQ /4/ AND GR 0...	for those members of authorized complements which are jobs not to be changed (but not for units),
...RESET...	use the authorized complement as actual (see RESET).
...AUTH JOB CODE NOT GR 0 ...	For members of complements which are units as opposed to jobs,
...ADD REPETITION ACTUAL...	use all authorized complement data except,
...SALARY = /2/ BACTUAL...	get budgeted salary from the totals of next lower-level units.
...CHANGE BACTUAL...	Update the budget and quantity totals for this unit from its newly created actual data.

**RESET**

LET RESET MEAN (ADD REPETITION ACTUAL (JOB CODE = /1/AUTH JOB CODE,  
TITLE = /1/AUTH TITLE,  
QUANT = /1/AUTH QUANT,  
SALARY = /1/AUTH SALARY)).

RESET uses any one of THISUNIT, SUBUNIT, SUBSUBUNIT as a parameter. RESET is used to create a repetition of "actual" job-type data by transferring material from the "authorized" data properties of the specified entry (which is presumed to be the entry selected by the ALTER phrase of the message).

Thus:

...RESET SUBUNIT...

copies authorized data into actual data for the authorized units of the entry being processed. RESET is used within RECOMPUTE.

**SUBSUBUNIT, SUBUNIT, THISUNIT**

LET SUBSUBUNIT MEAN (ORG(SUBUNIT AUTH UNIT CODE)).  
LET SUBUNIT MEAN (ORG(AUTH UNIT CODE)).  
LET THISUNIT MEAN ( ).

The three terms above signify sets of entries in the ORG file as specified in the Authorized Complement of a unit or set of units in another entry in the ORG file. They are used as parameters for RECOMPUTE. Statements using RECOMPUTE name a department (e.g., FOR ORG 2100---) and SUBUNIT refers to the set of units in the Authorized Complement of the department; i.e., those units which report to units which report to the department. THISUNIT refers to the department itself and is blank since it does not require any cross-entry reference.

**LEVEL0, LEVEL1, LEVEL 2**

LET LEVEL0 MEAN (THISUNIT SUBUNIT)  
LET LEVEL1 MEAN (SUBUNIT SUBSUBUNIT)  
LET LEVEL2 MEAN (SUBSUBUNIT THISUNIT)

RECOMPUTE actually requires two associated unit selection parameters, the second of which is determined by the first; e.g., if the first is SUBUNIT, the second is always SUBSUBUNIT. In order to make the specification of the pair simpler, three substitutions are used: LEVEL0 corresponds to department, LEVEL1 to units which report to it, and LEVEL2 to units which report to LEVEL1.

## APPENDIX B - FABLE

The FABLE Language, as an experimental on-line message language, is very rich in structure; only a subset of it will be described here.

### Basic Statement

A Basic Statement consists of three parts:

selection-part	action-part	output-part
----------------	-------------	-------------

as in:

FOR PERS.	IF LEVEL EQ CHIEF,	PRINT NAME,TITLE,UNIT.
selection	action,	output.

Either the action-part or the output-part may be omitted.

### Selection-Part

The Selection-Part is in the form:

FOR file-spec ALTER file (indirect-entry-spec).

as in:

FOR PERS ALTER ORG(UNIT).

which says:

" Select the PERS file and iterate through all its entries. Select those entries of the ORG file which correspond to values of the property UNIT in the PERS file."

The word FOR and its file specification initiate an iteration through all entries in the file. A list of entries or a single entry may also be specified. The FOR file is the source of property values used in the action part of the message, unless a property reference explicitly specifies otherwise.

The word ALTER and its file specification initiate an iteration through entries in its file specified by the current value of a property in the FOR file. The ALTER file is the only file in which changes, additions, or deletions may be made by the message, and is the source of property values for the output part. The use of the ALTER file is optional; if missing the FOR file assumes its functions.

An output-part produces an output file, which if named in a name-phrase will be saved for later use, otherwise it will be deleted immediately. The word SAVE instead of PRINT suppresses printing.

A FABLE message consists of any number of statements immediately following one another. It is terminated by an end-of-message character (which is produced by a key on on-line keyboard devices and for off-line inputs is supplied by the input-program in response to a terminator card).

APPENDIX C - FORMAT DESCRIPTION FOR FORMAT F1

The format is written in the SMAC macro language for the IBM 7030. Each statement begins with a call to the MOF macro, the Macro for Output Formats. The ' begins comments. Any number of expressions may be written on one MOF card.

MOF,BEGIN(F1),ROW,TL'	Names the format specification, declares it to be a standard "row-type" format, prints title (supplied in message).
MOF,*Q'	For each qualifying entry in the output file.
MOF,LIT(15)ORG UNIT'	Prints in 15 columns the literal words ORG UNIT
MOF,Q,/,N(15),V,/(2)'	Prints the qualifying object name (i.e., UNIT NO.). Skips to second line, prints a property name (will be REPORTS TO) and a property value for it, skips to second line.
MOF,N(15),V,/(2)'	Prints a name (ORG NAME) and a value (the name), skips to second line.
MOF,*L(2)'	Begins a two-pass loop through the following expressions, once for authorized, one for actual.
MOF,N,/(2),*L(5),N(0),L*,V(0)'	Prints the next property name (AUTHORIZED or ACTUAL complement) but suppresses printing of next 5 property names (JOB CODE, UNIT CODE, TITLE, QUANTITY, SALARY).
MOF,S(6),V,S(11),V(20),S(1)' MOF,V(2),S(6),V(6)'	Completes printing of property values within the repeating group and of totals.
MOF,/(2),L*'	Spaces to second line, terminate two-pass loop.
MOF,S(21),N(21),S(16),V(7)'	Prints next property (DEVIATIONS) if any.
MOF,Q*,END'	Terminates loop through qualifying entries and ends format specifications.



APPENDIX D - THE SKILCODE SUBROUTINE

A DAMSEL routine which takes as input a skill code, performs a table lookup, and delivers as output a skill name.

\$	ROUTINE SKILCODE'	NAMES ROUTINE,
\$	INPAR SKILNO, INTEGER'	SKILNO IS AN INPUT PARAMETER
\$	OUTPAR, SKIL, STRING'	SKIL IS AN OUTPUT PARAMETER
\$ START	ENTER'	ENTER INITIALIZES
,	LX, I, SIZE '	BLANK COL 1 SIGNIFIES ASSEMBLY
,		LANGUAGE, WHICH IS 7030 STRAP.
,		A ' IN COL 1 SIGNIFIES COMMENT
,		CAR. THE VARIABLE I IS AN INDEX
		REGISTER.
\$ MORE	IF EQ(P.SKILNO,SKILCODE(I)	
*	THEN FOUND'	* IN COL 1 IS CONTINUATION CARD
	V + I, I, 2.	ADD 2 TO INDEX VALUE
,	CB, I, MORE'	CB IS THE 7030 "COUNT AN INDEX
		REGISTER AND BRANCH" INSTRUCTION
\$	EXIT (ERROR)'	ERROR IF SKILCODE NOT FOUND
\$ FOUND	P. SKIL = NAME (I)'	P. IS NOTATION FOR PARAMETER
\$	EXIT	
\$ SKILCODE	INTEGER, 0110'	TWO CODES SHOW, OTHERS
\$ NAME	STRING, ADMIN, DIVISION'	WOULD BE ADDED BELOW
\$	INTEGER, 0130'	
\$	STRING, ADMIN, DEPARTMENT'	
	.	
	.	
	.	
SIZE	XW, 0, n'	COUNT OF NUMBER OF CODES
	MEND, 0'	ENDS PROGRAM

## COLINGO QUESTIONS AND ANSWERS

### QUESTION:

How long did it take you to get the answers from the time you got the documents until you got the final report?

### ANSWER:

I received the document about three weeks ago. Man hours? Let's see - it was myself for three weeks working on the first part of COLINGO itself and of computer time I think I used about a total of 2 or 3 hours before I finally debugged my own queries, etc.

### QUESTION:

How long does it take to train a person in COLINGO?

### ANSWER:

I would say about 3 or 4 days and a trainee could sit down at the console if he wanted to and type in a query and be able to get answers. Actually, most of the names of the system are in your language.

## MARK III FILE MANAGEMENT SYSTEM

Presented by: John A. Postley  
Director, Advance Information Systems  
Informatics, Inc.

The Mark III File Management System is one of the series of products resulting from a continuing project at Informatics which seeks to develop increasingly general-purpose file management systems for the solution of an almost unlimited variety of data base problems. The particular Mark III system is implemented on very modest equipment. Other systems currently under development at Informatics for third-generation equipment are correspondingly more flexible and efficient. The Mark III system requires a 1401 computer with minimum 12K core and 4 tape drives. Such equipment at SDC was used for the actual solution to the data base problem. The system also operates on the 1460, 1410, a 7010, and System/360 with 1401 emulator. The data base problem was also solved using Mark III on a System/360 with identical results obtained.

In approaching the problem, certain policy decisions were made by Informatics. The first such decision was to use the Mark III system as representative of the file management approach to this problem. Also as a matter of policy, no programming of any kind was carried out to solve this problem, although the Mark III system does provide the capability for writing its "own code" for this purpose. Finally, Informatics accepted the problem precisely as posed without attempting to redefine the problem in any way in order to improve the match between the solution and the "customer's" needs.

Three functions must be performed by the (not necessarily computer-oriented) user to solve the data base problem. The first is the creation of the file dictionary. The second is the creation of the input dictionary. And the third is the completion of the request forms. Any number of additional requests can be processed without repeating the first two functions.

### FILE DEFINITION

The first step in the problem solution was to lay out the file or files which are needed. Figure 1 illustrates the file layout selected. This file layout incorporates into a single file all information from both the organization file and the personnel file of the data base provided by SDC. There is one record for each of the 29 organizations in the Mark III file. Each record is of variable length and contains information about that organization, its sub-units, and all employees reporting to that organization.

The record is organized into four levels. Level 1 contains the basic description of the organization itself; each representation of Level 2 contains one of two kinds of information. The first kind of information describes the sub-unit, if any, reporting to the master organization. The second kind of

Form 124-6599-0  
Printed in U.S.A.MARK III FILE MANAGEMENT SYSTEM  
INTERNATIONAL BUSINESS MACHINES CORPORATIONMULTIPLE-CARD LAYOUT FORM  
TAPE FILE LAYOUT FORM

Informatics

Company SDC Data Base Problem by John A. Postley Date 9-20-65 Job No.

Sheet No.

LEVEL	Org Unit Code	Rps to Code	Organization Name Nmb of Sub Units or Pos. Titles	Ct	Auth. Annual Salaries	Sub Unit Name or Position Title	Sub Unit Code	Job Code	Empl. No.	Employee Name	Level	Current Annual Salary	Birth Date	Ct.	Primary Skill (322)	Secondary Skill (425)	Name
1	101	102	103	104	105	209	207	208	209	210	211	212	318	319	320	321	322
2	101	102	103	104	105	209	207	208	209	210	211	212	318	319	320	321	322
3	101	102	103	104	105	209	207	208	209	210	211	212	318	319	320	321	322
4	101	102	103	104	105	209	207	208	209	210	211	212	318	319	320	321	322

Figure 1. Tape File Layout Form

information describes each personnel position reporting to the master organization. Level 3 of the file contains employee information. One representation of the information at the third level exists for each position title described at the second level. Level 4 contains one group of information for each secondary skill attributed to each employee whose information is recorded at the third level.

Generally speaking, subrecords at the second, third and fourth levels are repeated subrecords. That is, one representation of the information shown in Figure 1 is incorporated in the record for each entity designated. The Mark III File Management System has the capacity to store records whose total length, including all repeated subrecords, does not exceed about 3,000 characters in a 12K machine or about 7,000 characters in a 16K machine. This capacity was nowhere near approached in the solution to this problem.

This file organization used is shown schematically in Figure 2. The subrecords in the file are segmented into fields containing the data required by the problem. This segmentation is illustrated in Figure 1. These fields and subrecords are defined to the generalized file management system by means of a file dictionary. The file dictionary describes for each field a number of pertinent facts. These include:

- The field number, which in the Mark III system is a three-digit, numeric designation.
- A code indicating whether the field is contained in Level 1 or in Level 2 or below.
- The field location relative to the beginning of the record for Level 1 fields or relative to the beginning of the subrecord for fields at Level 2 or below.
- The length of the field.
- The field number of the first field in the subrecord which contains this field.
- An edit code which defines the print format.
- The column heading which will appear automatically any time this field is printed on a report.

Fields 104 (number of Sub-Units or Position Titles), 211 (number of employees), and 319 (number of secondary skills) are file dictionary counters automatically developed by the system. They reflect the number of subrecords existing in each case at the lower level. Any size counter of one or more characters may be defined.

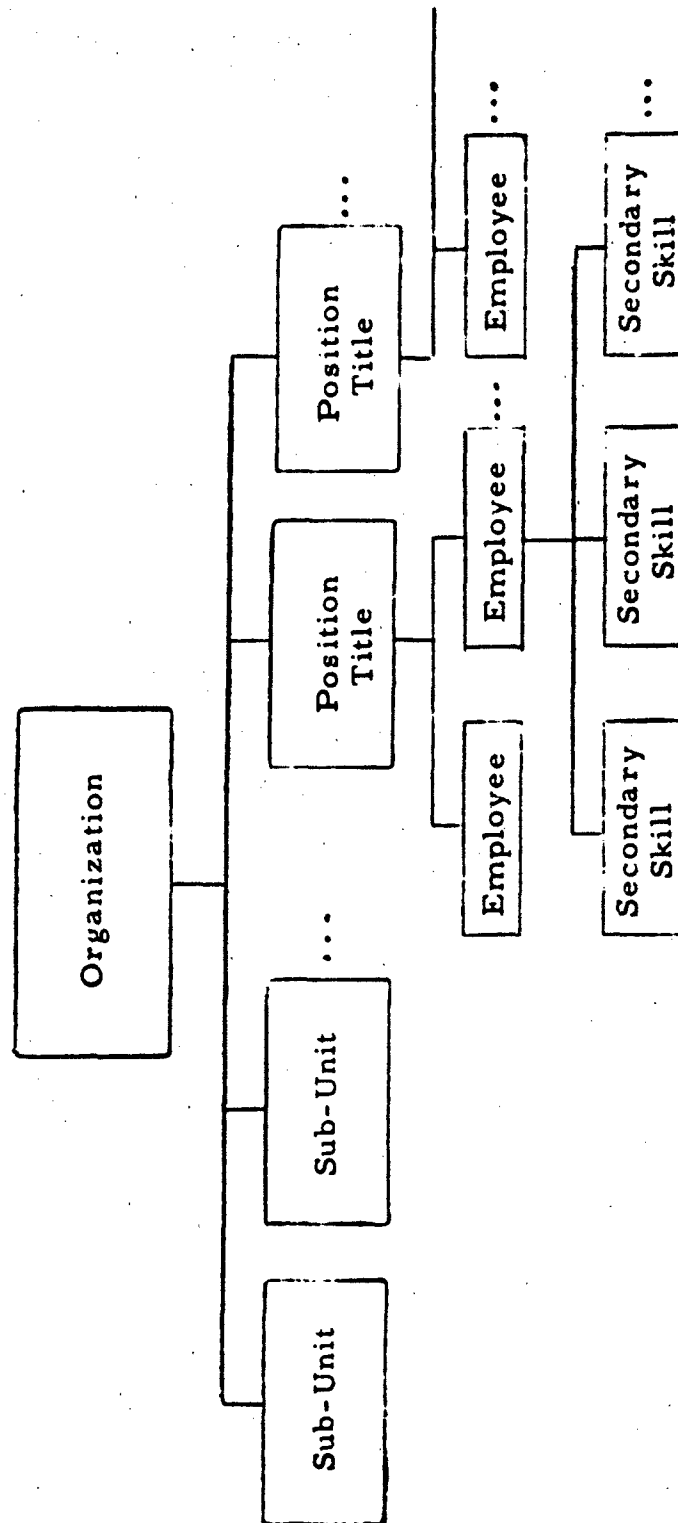


Figure 2. Schematic File Layout

## INPUT DEFINITION

The next step in the solution of the data base problem was to create the file, defined by the file dictionary, from the cards provided by SDC. For this purpose, an input dictionary must be defined for these cards. In this problem there are four cards in the input dictionary; one card corresponding to each level of the file as illustrated in Figure 3. However, in general, the system permits any number of cards to be used to form any given level of a file.

A considerable amount of flexibility is provided by the input dictionary. This flexibility permits almost any cards of any format to be used to create any file defined in the Mark III system. The input dictionary defines the cards which are actually at hand. There is one entry in the input dictionary for each input card, thus a total of four entries are recorded in the data base problem. The Mark III File Management System permits input information to be read either from cards or from magnetic tape although the latter is not required in this problem. Information contained in the input dictionary includes:

- Transaction code definition. Any existing transaction code may be used to identify each card transaction. From one to seven characters are permitted. The dictionary defines each code in terms of its length and its meaning. If no code at all is present in the cards, then the absence of a code can be defined in any one of these same ways. Possible meanings include:
  - adding a new record to a file
  - deleting a complete record from a file
  - inserting a new subrecord into an existing record
  - replacing the contents of existing fields with new values
  - replacing existing fields with blanks
  - summing algebraically and subtracting algebraically.
- The input record code distinguishes among the four cards which the system will see. Up to three characters of any description can be used for this purpose.
- The location on the card of the transaction code and the input record type is defined in the input dictionary.
- An indication of whether the item is used as an identifier or is simply an item of data.
- The position on the input cards of each item of data corresponding to the field in the file definition.

TM-2624/100/00

SR	Org Unit Code	Rpts To Code	Organization Name	Auth. Annual Salaries	Auth. Annual Salaries	Position Title	Job Code	Sub Unit Code	Empl. No.	Employee Name	Level	Current Annual Salary	Birth Date	Code	Name	Primary Skill
1	1 A	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2 A	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3 A	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4 A	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5 A	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6 A	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7 A	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8 A	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9 A	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
10	10 A	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	11 A	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
12	12 A	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
13	13 A	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
14	14 A	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
15	15 A	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
16	16 A	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
17	17 A	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
18	18 A	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
19	19 A	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
20	20 A	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
21	21 A	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
22	22 A	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22
23	23 A	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
24	24 A	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
25	25 A	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
26	26 A	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26
27	27 A	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27
28	28 A	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
29	29 A	29	29	29	29											

**Figure 3. Input Data Layout Form**



The entire process of establishing the file dictionary and the input dictionary is normally done only once for each problem. Occasionally, subsequent input conditions may arise in which case additional information must be entered into the input dictionary. In the present problem this complete process was performed only once and would not be expected to be carried out again. The entire process of creating dictionaries required about three or four hours of the time of an experienced analyst/programmer.

#### REQUEST PREPARATION

Having established the dictionaries for this problem, the next task was to define the applications which would result in the desired reports. For this purpose, requests were made by the system analyst. A request consists essentially of a set of two forms which constitute record-search specifications and retrieval and report specifications. The actual requests, as edited by the computer, are shown in Figure 4. No other information of any kind was required to produce the requested results. The total time used to enter this information into all requests was in the range of two to three hours.

In this problem the record-search specifications were relatively straightforward. In order to indicate to the system each separate logical search requirement, information is entered onto the record-search specification form indicating the name of a field, a comparator (equal to, greater than, etc.) and either the name of another field in the same record or an external value.

The Mark III capability for connecting such logical conditions (by and, and/or, and listing groups of conditions in up to 10 levels of parentheses) was not required in this problem.

Information required for the retrieval and report specifications form included, for each column of information desired, the name of the field to be printed in that column and an indication of whether that field was to control the sort, the totals, or itself to be an entry in a total to be computed by the system. In addition, when computations across a line are required, e.g., to compute the difference between actual and authorized salaries, the fields containing the actual salary must be designated, along with the appropriate arithmetic operator, on the line which designates the authorized amount.

The task of completing the requests, as described above, corresponds to the normal task of computer programming to solve the data base problem. In accomplishing this task two things are done. The first is the elimination of programming, replacing it with the information shown in Figure 4. The second is the complete elimination of documentation since the completed request forms themselves, along with the file dictionary, the input dictionary, and the existing documentation of the Mark III File Management System, completely define the problem at hand. Only the request forms, of course, need be created to solve this problem once the data base has been established by the system.

00115					
001201	316	G			
001301	201101	12	1		
001302	203526	10	2	T	
001303	202208	6	31		
001304	204211		4	T	
001305	205316	4		1	
001401					
001402					
001403					
001404					
001405					
001406					

x1101

CONTROL REPORT

SECOND SYMPOSIUM ON COMPUTER-CENTERED DATA BASE SYSTEMS  
SYSTEM DEVELOPMENT CORPORATION  
SEPTEMBER 20-21, 1965  
SANTA MONICA, CALIFORNIA

Figure 4. Input Content as Validated  
(Sheet 1 of 6)



X0331

	101	G	1
00315	20111		11
003201	20110		2
003302	201103		3
003303	201207		4
003304	204208		5
003305	206209		62
003306	207210		7
003307	208212		8
003308	209211		9
003309	210316		2
003310			
003401			
003402			
003403			
003404			
003405			
003406			
003407			
003408			
003409			
003410			
003411			

## GROUP REPORT

THIS REPORT SHOWS AUTHORIZED PERSONNEL COUNT AND AUTHORIZED ANNUAL SALARIES FOR EACH GROUP. IT ALSO SHOWS ACTUAL NUMBER OF EMPLOYEES AND CURRENT ANNUAL SALARIES FOR EACH SECTION, WITH SUB-TOTALS FOR THE GROUPS.

SECOND SYMPOSIUM ON COMPUTER-CENTERED DATA BASE SYSTEMS  
SYSTEM DEVELOPMENT CORPORATION

SEPTEMBER 20-21, 1965  
SANTA MONICA, CALIFORNIA

Figure 4. Input Content as Validated  
(Sheet 3 of 6)



0051						
005201	317	E	H	HEAD		
005202	0A315	N		1110		
005203	0A320	E		1110		
005204	1R320	E		1113		
005205	1R320	E		1113		
005206	1R423	F		1113		
005301	201314		1			
005302	202313		2			
005303	203101		5 3			
005304	204320		5 4			
005305	205321		5 5			
005306	206423					
005307	207424					
005401						
005402						
005403						
005404						
005405						
005406						

X0103  
X0103

# DEMAND REPORT NO. 2

SECOND SYMPOSIUM ON COMPUTER-CENTERED DATA BASE SYSTEMS  
SYSTEM DEVELOPMENT CORPORATION  
SEPTEMBER 20-21, 1965  
SANTA MONICA, CALIFORNIA

Figure 4. Input Content as Validated  
(Sheet 5 of 6)

X0451

0061	1.1	5	1
006201	0A211		
006202	201101	11	
006301	202102	2	
006302	203103	3	
006303	204203	42	
006304	205209	5	
006305	206210	6	1
006306	207212	7	1
006307	208211	8	1
006308	209316		2
006309			
006401			
006402			
006403			
006404			
006405			
006406			
006407			
006408			
006409			
006410			

## EXCEPTION REPORT

THIS REPORT SHOWS INFORMATION FOR ALL SECTIONS WHERE THE  
ACTUAL NUMBER OF PERSONS IS EQUAL TO OR GREATER THAN THE  
AUTHORIZED NUMBER OF PERSONS.

SECOND SYMPOSIUM ON COMPUTER-CENTERED DATA BASE SYSTEMS  
SYSTEM DEVELOPMENT CORPORATION  
SEPTEMBER 20-21, 1965  
SANTA MONICA, CALIFORNIA

Figure 4. Input Content as Validated  
(Sheet 6 of 6)

In order to avoid the need to retain the actual forms completed by the analyst, the system automatically edits each request as it is entered and prints out the edited request (Figure 4) as a permanent record before proceeding with the processing. Requests which fail to pass the extensive edit tests are flagged according to the errors committed and automatically withdrawn from further processing.

Two passes of the master file on tape were required to solve the complete data processing problem. One pass was required initially to create the master file from the cards provided. Updating was done in the same pass although a separate pass could have been employed. This task would normally not be repeated until the data in the data base changed. To produce the required results, one pass of the master file was required to retrieve the necessary information. One sort of all retrieved data (run simultaneously) was then performed, and one pass of the sorted retrieved data under control of the report program produced all of the required reports. Thus, for all requests, the data base itself was passed only once and the results were then sorted and reports processed and printed.

In order to illustrate the results of Mark III operation, Figure 5 shows a portion of the Section Report, and reproduces all of Demand Reports 1 and 2.



ORG RPTS UNIT TO CCCE CODE	ORGANIZATION NAME	JOB POSITION TITLE OR CODE-SUB-UNIT NAME	AUTH QTY.	AUTH. ANNUAL SALARIES	NO. OF PERSONS EMPLOYED	CURRENT ANNUAL SALARY
2111 2110	PROPOSAL SECTION	1110 CHIEF	1	\$ 14000	1	\$ 14000
2111 2110	PROPOSAL SECTION	1120 TECH ENGR	1	\$ 10500	1	\$ 10500
2111 2110	PROPOSAL SECTION	1130 ELEC ENGR	1	\$ 11500	1	\$ 11500
2111 2110	PROPOSAL SECTION	1330 DRAFTSMAN	1	\$ 8500	1	\$ 8200
2111 2110	PROPOSAL SECTION	1350 COST ESTIMATOR	1	\$ 8100	1	\$ 8100
			5	\$ 52600	5	\$ 52300
2113 2110	ADV. SYSTEMS SECTION	0110 CHIEF	1	\$ 13000	1	\$ 13000
2113 2110	ADV. SYSTEMS SECTION	1110 SYSTEMS ENGR	1	\$ 12000	1	\$ 12000
2113 2110	ADV. SYSTEMS SECTION	1120 TECH ENGR	1	\$ 12000	1	\$ 12000
2113 2110	ADV. SYSTEMS SECTION	1130 ELEC ENGR	1	\$ 12000	1	\$ 11000
			4	\$ 49000	4	\$ 48000
2115 2110	PROD. SPEC. SECTION	1110 CHIEF	1	\$ 12000	1	\$ 12000

Figure 5. Illustration of Mark III Operations  
(Sheet 1 of 3)

1 December 1965

3-138

TM-2624/100/0

Q.NO. 004				DEMAND REPORT NO. 1			
EMPLOYEE NAME	EMPL NO.	ORG UNIT CODE	ORGANIZATION NAME	PRIM SKIL CODE	PRIMARY SKILL	SEC SKIL CODE	SECONDARY SKILL
BOYC, W. V.	15052	2135	MODEL SHOP SECTION	1351	MECH TECHN		
CATES, C. L.	81130	2313	FAB. SECTION B	3340	MACHINE OPR		3340 MACHINE OPR
FLETCHER, M. W.	21475	2133	PROD. ENGR. SECTION	1365	TOOL DESIGNER		3345 MILLING MACH OPR
JORTON, R. A.	17590	2313	FAB. SECTION B	3340	MACHINE OPR		3360 HEAT TREAT OPR
GREEN, S. D.	34840	2313	FAB. SECTION B	3340	MACHINE OPR		3340 MACHINE OPR
LARNER, L. F.	78885	2353	EQUIP. MAINT. SECTION	7120	MAINT ENGR (E)		3345 MILLING MACH OPR
LEVITT, P. S.	32924	2311	FAB. SECTION A	3340	MACHINE OPR		3360 HEAT TREAT OPR
CITILE, E. F.	42055	2311	FAB. SECTION A	3340	MACHINE OPR		3345 MILLING MACH OPR
MILLER, F. L.	52467	2311	FAB. SECTION A	3340	MACHINE OPR		3360 HEAT TREAT OPR
						3550	EXPEDITOR
						3570	DISPATCH ASSY
						7320	MAINT TECH (E)
						3510	ASSEMBLER

16 ITEMS RETRIEVED FOR THIS REQUEST

Figure 5. Illustration of Mark III Operations  
(Sheet 2 of 3)

1 December 1965

3-139

TM-2624/100/00

DEMAND REPORT NO. 2

REC.NO. 005

EMPLOYEE	EMP NO.	ORG UNIT CODE	PRIN SKIL CODE	SEC SKIL CODE	SECONDARY SKILL
ARNETTE, L. J.	37113	2115	1130	ELEC ENGR	
ETTINGER, G. J.	13581	2113	1130	ELEC ENGR	
FRANCIS, G. L.	21777	2113	1110	SYSTEMS ENGR	1130 ELEC ENGR
HENDERSON, R. G.	18130	2111	1130	ELEC ENGR	
PASSARD, L. R.	36472	2132	1130	ELEC ENGR	
RAYCRO, H. F.	28654	2123	1130	ELEC ENGR	
SILCOTT, D. N.	10295	2132	1130	ELEC ENGR	
SKINNER, J. P.	89876	2132	1130	ELEC ENGR	
STADERMAN, P. K.	80823	2122	1130	ELEC ENGR	

9 ITEMS RETRIEVED FOR THIS REQUEST

Figure 5. Illustration of Mark III Operations  
(Sheet 3 of 3)

**QUESTION:**

What language is used in the queries?

**ANSWER:**

The language in the queries is not a language in a programming sense. It's a process of filling out a form--or two forms, to be precise. We make use of a glossary which gives a three-digit name to every field in the file. If, for example, you want the man's name that's 108, say: "I want 108." Then you place that request in the column on the form that indicates which field is desired and send it out. The order in which the requests are placed on the form indicates the order across the page in which the answers will be printed. If you want to total that field in some field like authorized salary, a one-digit code indicating the one you want is entered. If you want to multiply, add, or whatever, one field by or to another, you put the multiply sign, and the name of the other field that you want to multiply, by this one which is already on the line. The process is really concerned with filling out forms and is not in the programmer sense at all.

**QUESTION:**

Does the form enable referral of a later line to a previous line?

**ANSWER:**

We do have that capability where it is appropriate. It's appropriate when you want to, for instance, multiply two columns together or multiply the fields in two columns together. In fact, the only way you can accomplish the multiplication is by referring to another line that has to be identified. If, for instance, you say, "Multiply this field by another field," and you've failed to identify the other field anywhere, the system won't do it, and it will tell you before it even runs the report that you have done something wrong. We haven't gone very far with that because so little information has to be entered in the first place. Basically, the request for the problem today took us between two and three hours to fill out the forms for all of the requests in this problem--we had a lot of other time for beautifying it.

**QUESTION:**

How complex can the criteria be--combinations of "and's" and "or's"?

**ANSWER:**

Well, it's far more complex that anyone can write, including me. You can have essentially an unlimited string of "and's" at any time connected to an unlimited string of "or's." You can then nest those in parentheses and combine

them with more "and's" and "or's," etc. You can nest up to ten levels of parentheses, and nobody can write a request that's anywhere near that complex.

QUESTION:

There's probably some limit on these queries?

ANSWER:

Yes, but it has to do with space and not complexity.

QUESTION:

The figure showed several types of inputs. Does this imply you can merge tape-card inputs, etc?

ANSWER:

Yes, if you have enough tapes, for one thing. With the minimum configuration of four tapes, you can't do it. You have to have five tapes to do that. Also, several people have asked us about the capability of retrieval from multiple files. As the system stands now, it doesn't do that. It retrieves only from one tape file at a time--any total number, but one at a time. However, we find this an almost trivial modification which we just haven't made yet. We do have the capability to merge cards off-line and then read them in.

QUESTION:

On the updating transaction--if the file is organized by unit code with the three transactions accessed by personnel employee number, do you search the whole files?

ANSWER:

No, we did not have to bring it in by unit code. The system will look at the third level to find one employee, and it did, in fact, do just that. Normally the user doesn't have to even know how the file is organized. If he's looking for a field, the system looks in the dictionary to find that the field employee number is at the third level of the file (level of the record). It then goes down to the third level of the record and looks at all those until it finds that employee.

QUESTION:

So, in essence, you saved a table of the transactions, and as you went through, the file would look at each employee and see if this was the one that has had a transaction?

1 December 1965

3-142

TM-2624/100/00

ANSWER:

Yes.

QUESTION:

Don't the transactions have to be in the same sequence as the master file?

ANSWER:

You're referring to the updating transactions? Yes, they do.

ON-LINE DATA MANAGEMENT SYSTEM FOR THE  
MASSACHUSETTS GENERAL HOSPITAL\*

Presented by: Paul A. Castleman  
Director, Client Service Group  
Bolt Beranek and Newman Inc.

THE HOSPITAL TIME-SHARING SYSTEM

There is a time-shared computer running in Cambridge, Massachusetts, servicing 48 remote teletypes. These teletypes are connected to the computer 24 hours a day and provide a variety of functions ranging from an on-line programming system and a JOSS-like algebraic interpreter to a hospital communication and information-retrieval system.

The principal function of the computer system is to investigate the feasibility of using a computer to handle the real-time medical-administrative operations of a hospital. It is being used presently by three categories of users. To a group of programmers at Bolt Beranek and Newman Inc., in Cambridge, the computer is available for on-line preparation and debugging of programs used by the computer layman at Massachusetts General Hospital in Boston. At the hospital, the computer is being used as an interpretive communication system, presently in parallel operation at two nursing stations and in two laboratories. And a group of administrative and research members of the hospital staff are using a broadly applicable data input and retrieval system for their particular project needs. This group of programs, to be distinguished from the programs designed for specific hospital functions, was used in solving the data base problem.

The hardware configuration of the computer system includes a modified Digital Equipment Corporation PDP-1 computer, a 50-million-character UNIVAC Fastrand drum, two magnetic tape units, and a small 400-thousand-character drum. About half of the 24,000-word computer core is used for the Time-Sharing Executive routine. The rest of the core contains special-purpose interpreters, subroutines, and 4,000 words for the running user program itself. The small drum is used for program storage during multiprogramming; the big drum and magnetic tapes are used for bulk storage of data and library programs (Figure 1).

The system has been developed as a research project and is not designed for large-scale production. For example, 4K user programs, small buffers, 10 character-per-second teletype output, and one-tenth of a second drum access all preclude such production.

---

\*Work reported here was performed under contract PH43-62-850, and Grant GM 00263-01 with the National Institutes of Health, U.S.P.H.S., and under a grant from the American Hospital Association.

Since the system is designed for use by the nonprogrammer, careful human engineering of user programs was necessary. All user programs appear to hospital users in the form of a question-answer type of dialogue and provide abundant checking of both semantic and syntactic errors, error messages, facilities for correcting errors easily, and rapid verification of encoded entries.

A major effort was also made to provide enough flexibility to accommodate modifications necessary to suit the changing techniques and needs of a hospital. Because the programming system runs at the same time as the hospital functions, the hospital may be using one program while a programmer is preparing, assembling, and checking out another program, and putting it in the library. As a result, the turnaround time for programming modification can be very short--a matter of minutes or hours.

The basic procedure for using the system is as follows: from a teletype terminal a user calls the desired program from the library. From then on the user carries on a dialogue with the computer, with the user-program acting as intermediary. The course of the dialogue, as well as the actual functions performed, are determined by the program, the user, and the current data.

#### THE ON-LINE DATA INPUT AND RETRIEVAL SYSTEM

##### The Programs

The data input and retrieval system may be described in terms of three basic information-handling operations. First, there is the description of the format and structure of the data; second, the assimilation of the data into the computer (according to that description); and third, the actual retrieval of the data stored in the computer.

The basic logical unit of data to be referenced at retrieval time is called a "field" (e.g., employee number, birthdate, skill); the various configurations that the field can assume are called "field values" (e.g., 234345, 01/01/1964, mechanic). The principal function of the description program is to let the researcher describe the fields, their legitimate values, and their location on input cards (if those are used).

A complete description of a field consists of its text name, its position in the tree structure of the record (discussed below), the type of information it may contain (integer, floating point, date, or text), the location of data on the card input, and, finally, a syntactic definition. Since field values are grouped together into one record (episode or item), there is some field or group of fields which unites cards of a record and distinguishes one record from all the others. This field might be employee number or organization unit number. Thus, another function of the description is to permit the operator to define the field which is to identify the record.



There are two programs used to put data into the file; one accepts data from card images on magnetic tape, the other accepts data directly from the teletype. The former is useful in large-scale inputting of data. For entering small files or for updating and modifying large ones, teletype input is more useful.

There is a general-purpose retrieval program which accepts any Boolean or arithmetic combination of fields as a selector. A linear search is performed on the file to find those records fitting the selector. A three-valued logic (true, false, and indeterminate) is used. The program can print specified fields from the selected records and/or two-dimensional tables of values. These tables or matrices may then be transferred to the algebraic manipulator for sophisticated statistical analysis.

Figure 2 illustrates the interaction of the programs with the file.

### The Language

The operation of the generalized data input and retrieval system is based on the answers to specific user program questions. There is no need for anything approaching a programming, or even a procedural, language. The answer, often a simple "yes" or "no," is stored as a parameter in the program and sometimes also in the file itself. The program interprets these answers and performs its functions immediately.

The execution of functions may be delayed and performed at repeated intervals in the future. The answers to the questions of the file description and retrieval programs can be thought of as coding for a compiler or interpreter. The analogy is strengthened by the fact that these "programs" can then be stored on the program library after the questions are answered, but before the execution of any function. The stored programs are, in effect, object programs. For example, a program to generate a particular kind of report can be "written" by first answering the retrieval program questions and then putting the entire program on the library.

### The File Structure

The file itself remains on the big drum. Each record in the file is variable in length, consisting of chained 150-character blocks. The logical internal structure of the record is that of a one-level n-ary tree (see Figure 3). Fields within the record may be either variable or fixed length. They can be numbers, dates (required for date arithmetic), codes\* or text strings. Several fields may be grouped in one node of the tree.

---

\*The field-value dictionary facility is not yet operational.

While the records form the actual hard data, there are several items of description and overhead in the complete file (see Figure 4). There is a format description of a record--i.e., a key to the structure of the records for this particular file. There is also a description of the various formats of cards which may be used to add data to the file. The specifications for each field are stored as separate dictionary items in the file. Finally, there is an ordered table of contents to the records. (The order of this table, which may be changed, determines the order of the printout at retrieval time and defines the order of a linear search.)

One way of describing in detail the workings of this input-output procedure is by example. On the following pages are schematic drawings, actual typescripts, and notes for the handling of the data base problem. Many of the features, as well as the shortcomings, of the system will be noted.

#### THE HANDLING OF THE DATA BASE PROBLEM

Two basic files are maintained--the employee file and the organization file (see Figures 6 and 9). These two files contain all the data of the problem, and any updating is performed on one of these files. Also, any retrieval which admits to the organization of these files (e.g., employee demand reports and control counts) can be performed directly on one of these two files (see Figures 12 and 15). The periodic report and its look-alike exception report need to consider data in both files and also to reflect a different set of levels for the employee data. That is, while the employee file is organized with one record for each employee, the periodic report necessitates the employee data organized around organization unit membership. Because the retrieval system can deal with only one file at a time, a merge of the two basic files into a third temporary file organized for the periodic report is necessary (see Figure 21).

The generation of this third file is also necessitated by the limitation in the depth of the tree levels. The structure of the data base is that of a multi-leveled tree. The required retrieval considers various sections of this tree. As was mentioned, our system only represents data as a two-level structure. The method used in retrieving from other levels is to restructure the file for the particular retrieval. The actual process consists of rewriting the file in a linear fashion (e.g., writing card images onto magnetic tape) and then re-assimilating the data into a new file of a different tree structure. This method was used in generating the periodic report (see Figure 19).

In generating the hypothesis report, our failure to represent data as a full four-level tree necessitates additional operations. When data for a draftsman in a section are changed, the corresponding figures in the associated group record must also be modified. Our file structure does not allow us to calculate both figures in one pass. The flow diagram (Figure 27) outlines a reasonable procedure for handling the hypothesis situation. This most direct approach

requires one clerical step. Simple arithmetic is necessary before entering the correct values in the group record.

#### THE SOLUTION TO THE DATA BASE PROBLEM

In the typescripts that follow, the characters underlined are the answers entered by the user. The other text is either the questions, comments, or actual retrieval output generated by the programs. The typescripts are grouped by the function performed; each group is preceded by a flow diagram. The diagrams of the files generated are also included. Where the precise meaning of the questions or answers is not clear, see Reference 2.

#### REFERENCES

1. G. O. Barnett, M.D., J. J. Baruch, Hospital Computer Project Memorandum No. Eight., Massachusetts General Hospital.
2. S. I. Allen, M.D. (ed.), Research Cycle Users' Manual, Bolt Beranek and Newman Inc.

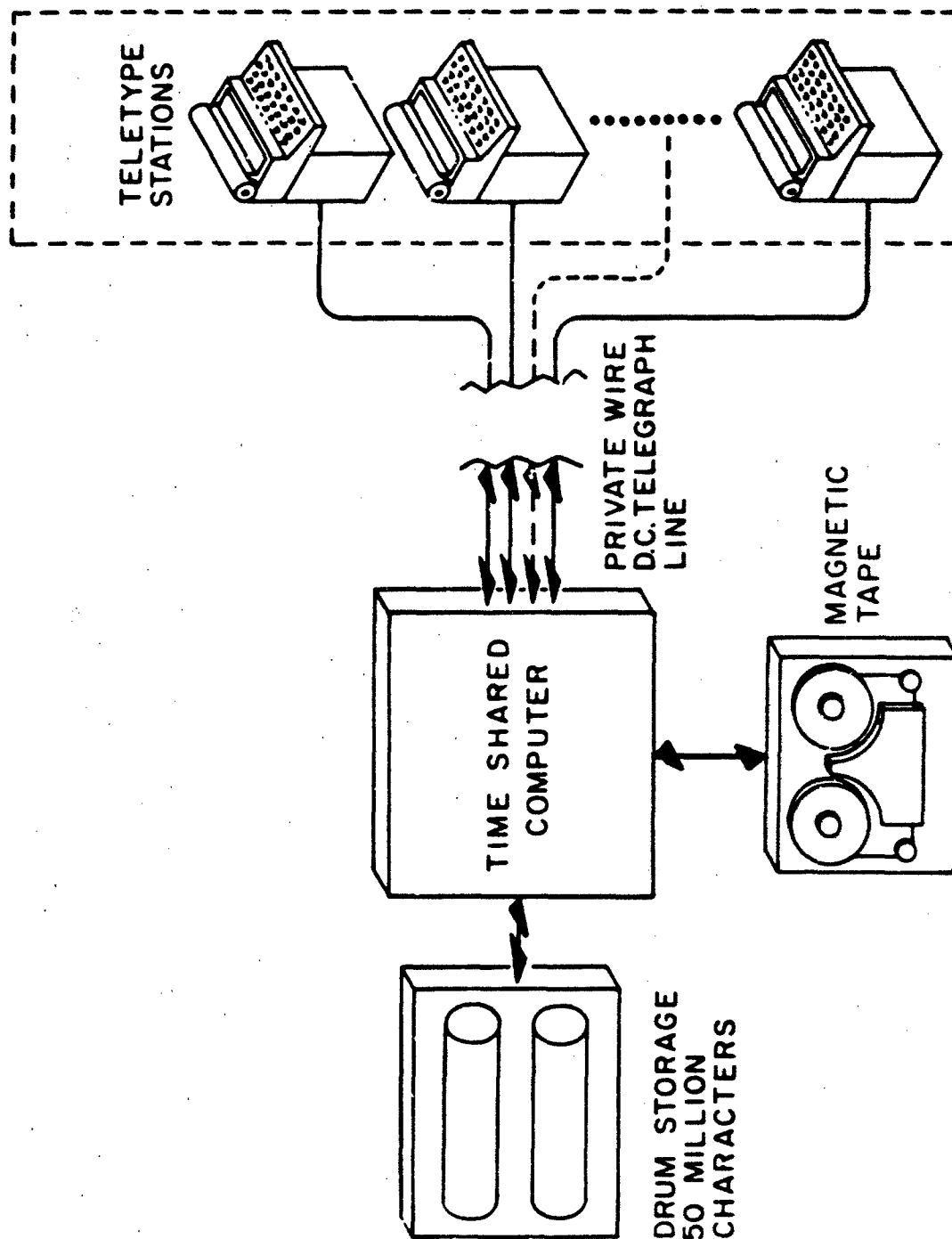


Figure 1. Computer-Based Communication System

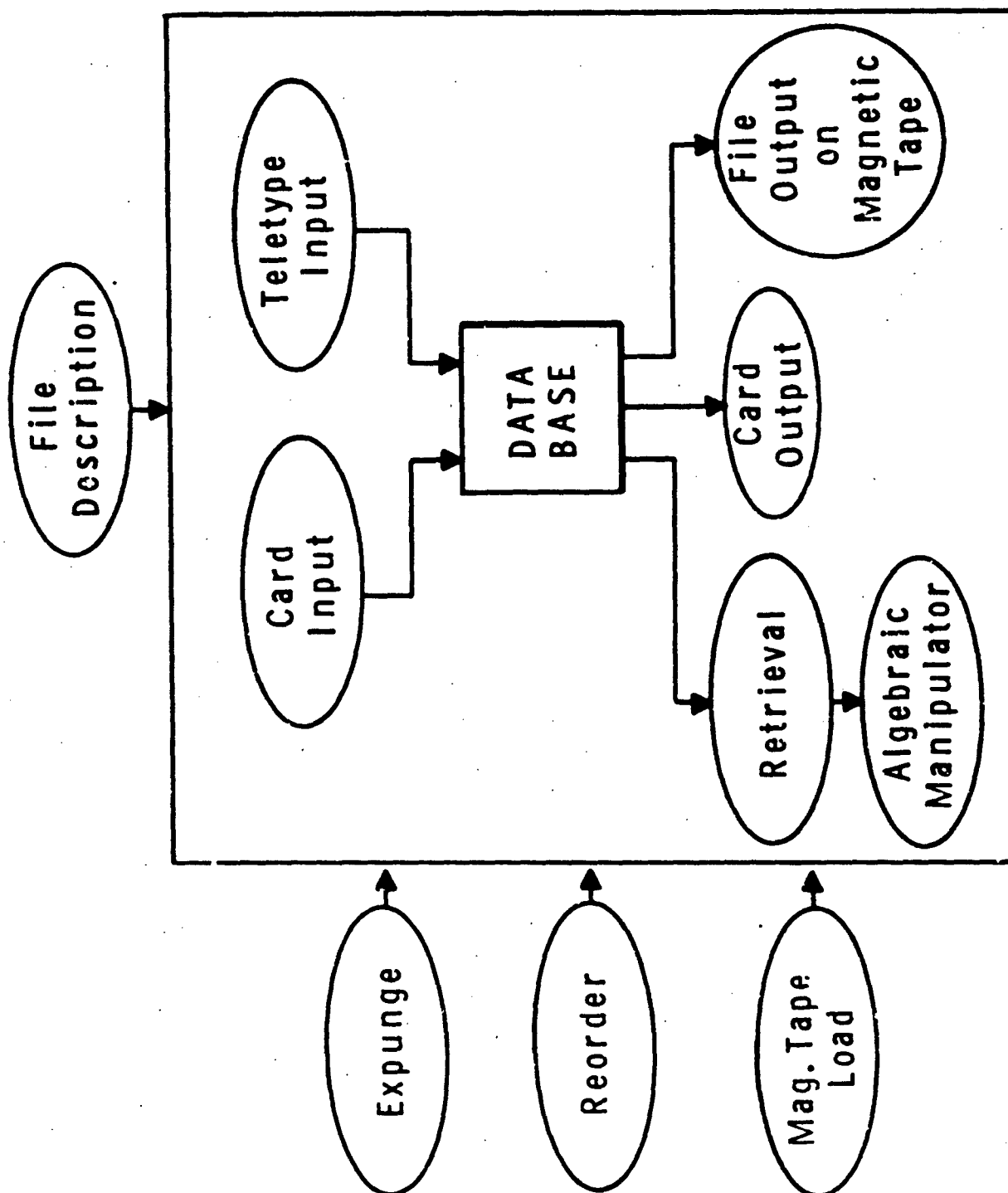


Figure 2. File-Handling Functions (User Programs)

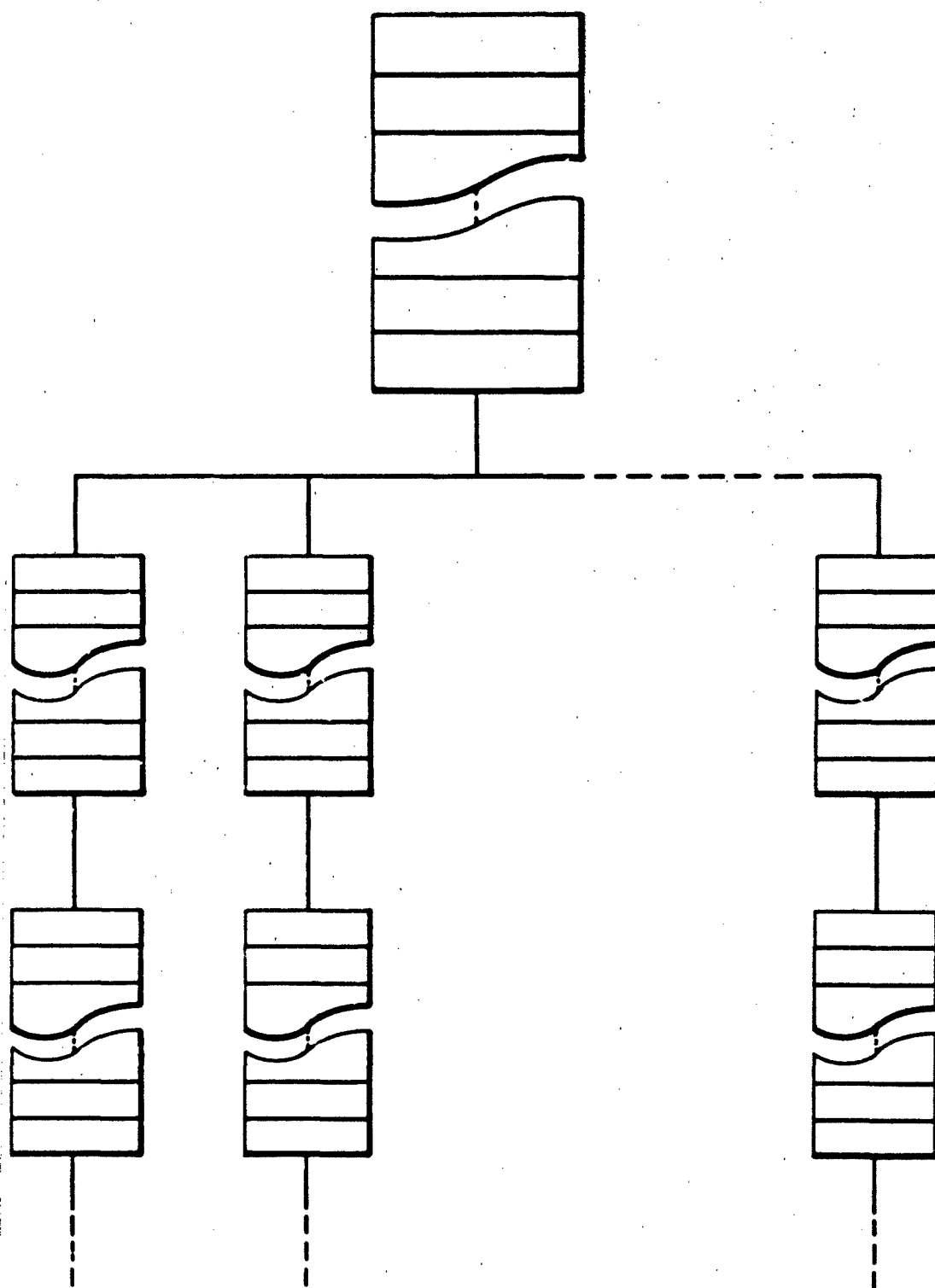


Figure 3. General Structure of a Record

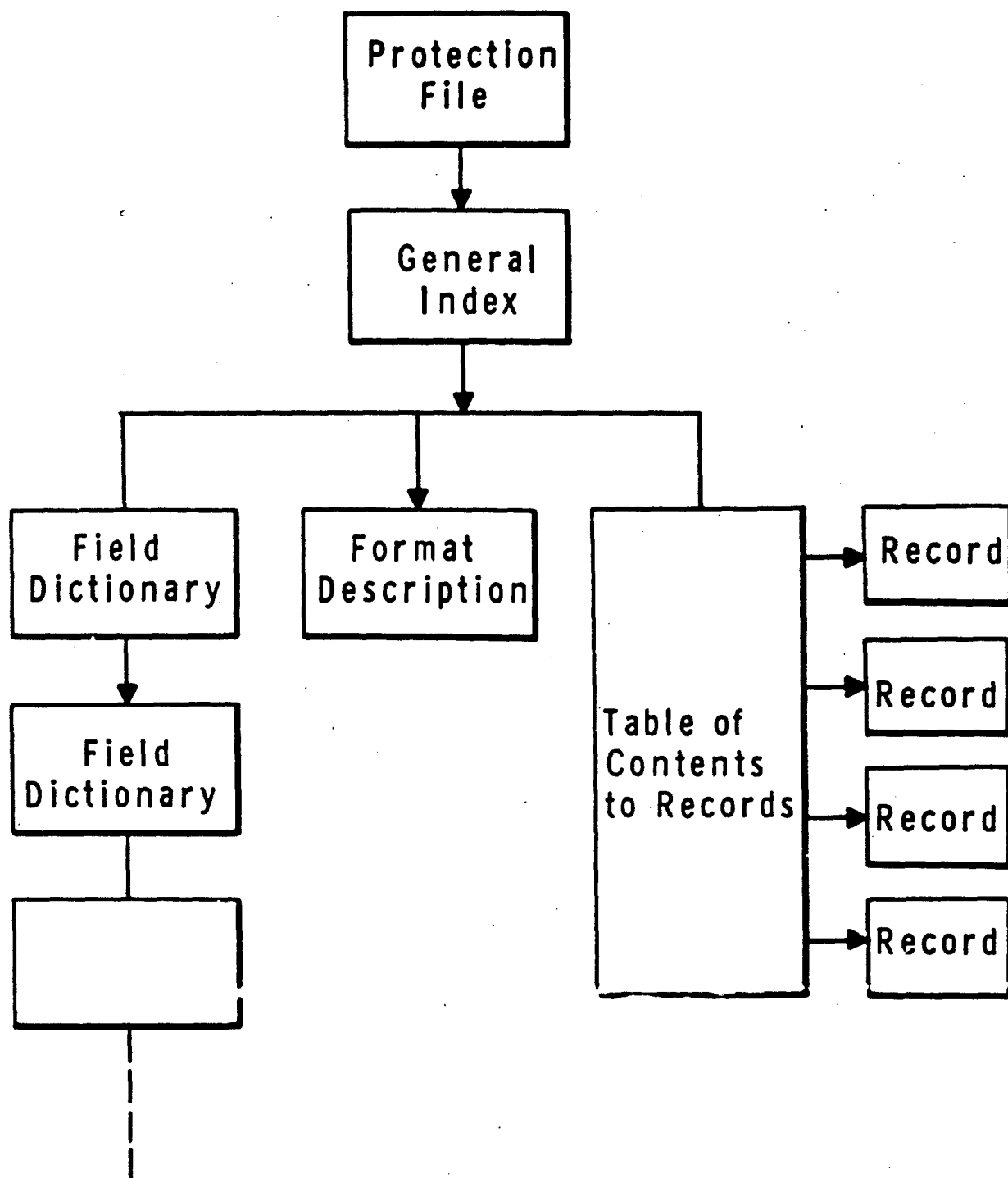


Figure 4. General File Structure

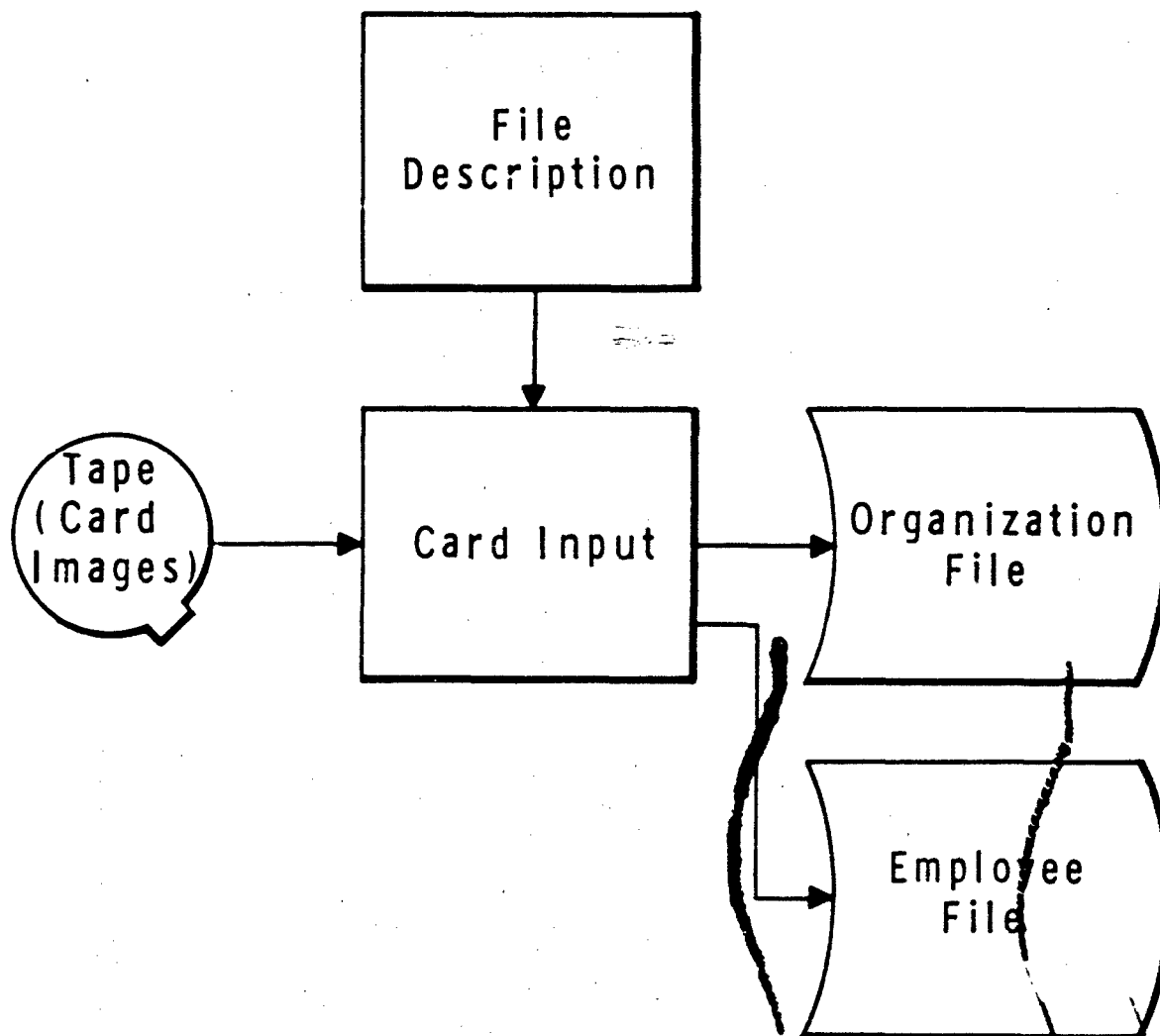


Figure 5. Create the File



## STRUCTURE OF A RECORD IN THE EMPLOYEE FILE

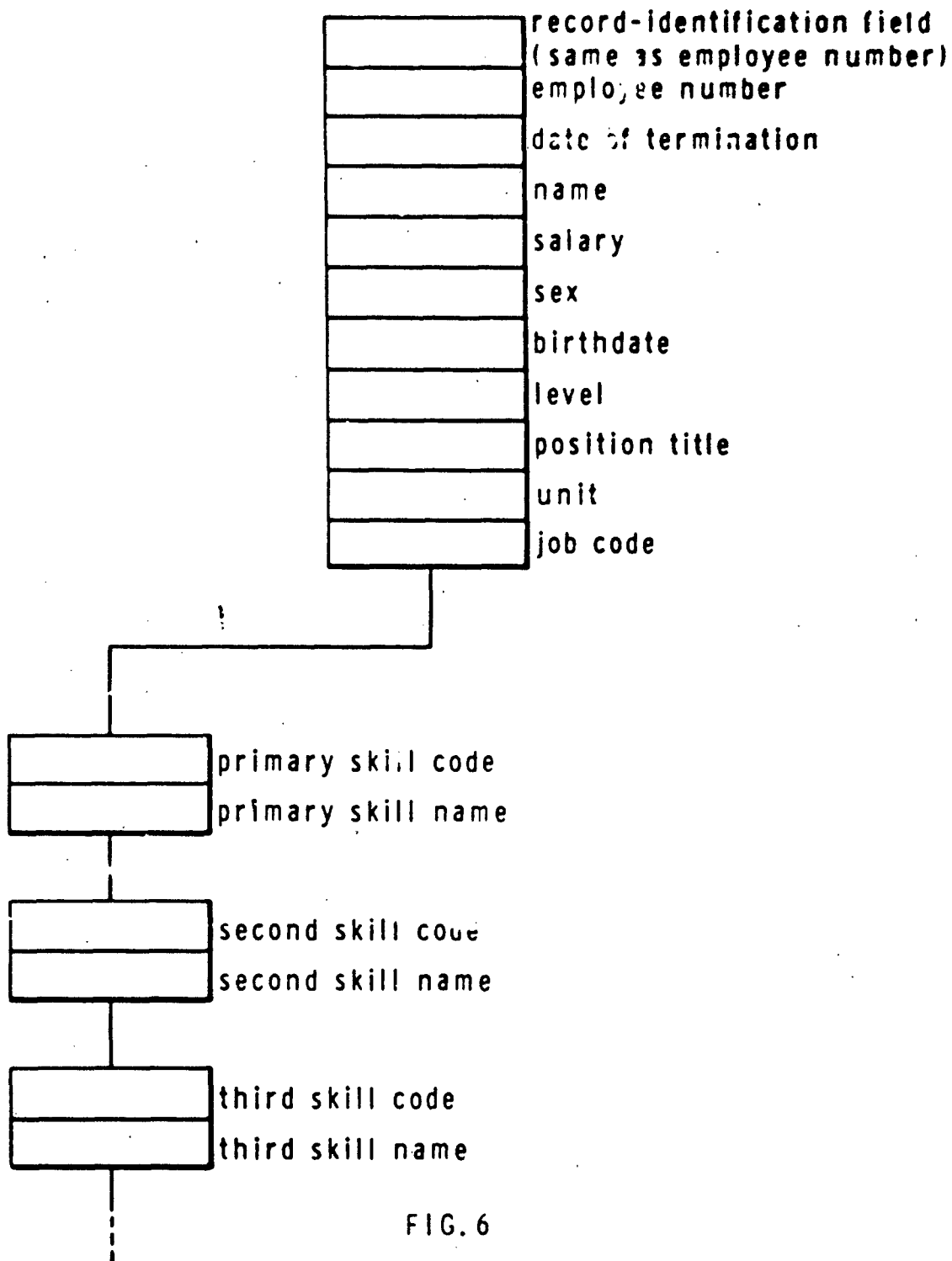


Figure 6. Structure of a Record in the Employee File

"G"  
RESEARCH FILE FORMAT DESCRIPTION

0.1 FILE TITLE: EMPLOYEE FILE  
0.2 CONFIDENTIAL CODE: SDC  
0.3 NO. OF CARD-TYPES: 3  
0.3.1 DEFINING COLUMNS FOR TYPE #1: 80  
0.3.1.1 SYNTAX DEF: "1"  
TRY ME: 1...OK.  
TRY ME: 2...NOT OK.  
TRY ME: -  
0.3.1.2 NO. OF CONTINUATION CARDS: 0  
0.3.2 DEFINING COLUMNS FOR TYPE #2: 80  
0.3.2.1 SYNTAX DEF: "2"  
TRY ME: -  
0.3.2.2 NO. OF CONTINUATION CARDS: 0  
0.3.3 TYPE #3 IS ASSUMED TO BE ALL THE OTHERS.  
0.3.3.1 NO. OF CONTINUATION CARDS: 0

1 FIELD #1 NAME: NUMBER  
1.1 UNIQUE TO RECORD? YES  
1.2 TYPE: REGULAR  
1.3 CARD TYPE: 1,2,3  
1.4 COLUMNS: 1-5  
1.5 SYNTAX DEF: 99999  
TRY ME: 33445...OK.  
TRY ME: 11220...NOT OK.  
TRY ME: -

2 FIELD #2 NAME: DATE OF TERMINATION  
2.1 UNIQUE TO RECORD? YES  
2.2 TYPE: DATE  
2.3 CARD TYPE: 2  
2.4 COLUMNS: 60-69  
2.5 CARD TYPE: -  
2.6 SYNTAX DEF: XXXXXXXXXX<10X>  
TRY ME: -

3 FIELD #3 NAME: NAME  
3.1 UNIQUE TO RECORD? Y  
3.2 TYPE: REG  
3.3 CARD TYPE: 1  
3.4 COLUMNS: 9-33  
3.5 CARD TYPE: <10X><10X>XXXXXX FIX  
3.5 CARD TYPE: -  
3.6 SYNTAX DEF: <10X><10X>XXXXXX  
TRY ME: -

4 FIELD #4 NAME: SALARY  
4.1 UNIQUE TO RECORD? Y  
4.2 TYPE: INTEGER  
4.3 CARD TYPE: 2  
4.4 COLUMNS: 16-21  
4.5 CARD TYPE: -  
4.6 SYNTAX DEF: 99999910000000  
TRY ME: 123456...OK.  
TRY ME: ...OK.  
TRY ME: -

Figure 7a. Research File Format Description

5 FIELD #5 NAME: SEX  
 5.1 UNIQUE TO RECORD? Y  
 5.2 TYPE: R  
 5.3 CARD TYPE: 2  
 5.4 COLUMNS: 9  
 5.5 CARD TYPE:  
 5.6 SYNTAX DEF: "M" "F" 10  
 TRY ME: -

6 FIELD #6 NAME: BIRTHDATE  
 6.1 UNIQUE TO RECORD? Y  
 6.2 TYPE: DATE  
 6.3 CARD TYPE: 2  
 6.4 COLUMNS: 10-15  
 6.5 CARD TYPE:  
 6.6 SYNTAX DEF: #####1991"/"991"/"[[1"18"["7"1"8"1"9"19]]1"19"99]  
 TRY ME: 010165...OK....FILED AS: 01/01/1965  
 TRY ME: 010185...OK....FILED AS: 01/01/1885  
 TRY ME: 022965...OK. BUT ITS NOT OF THE TYPE INDICATED.  
 ...FILED AS: 02/29/1965  
 TRY ME: -

7 FIELD #7 NAME: LEVEL  
 7.1 UNIQUE TO RECORD? Y  
 7.2 TYPE: R  
 7.3 CARD TYPE: 1  
 7.4 COLUMNS: 42-45  
 7.5 CARD TYPE:  
 7.6 SYNTAX DEF: XXXX  
 TRY ME: -

8 FIELD #8 NAME: POSITION TITLE  
 8.1 UNIQUE TO RECORD? Y  
 8.2 TYPE: R  
 8.3 CARD TYPE: 1  
 8.4 COLUMNS: 46-75  
 8.5 CARD TYPE:  
 8.6 SYNTAX DEF: <10X>10X>\\\\\\\\<10X><10X>  
 TRY ME: -

9 FIELD #9 NAME: -8.6  
 8.6 SYNTAX DEF: <10X><10X><10X> -OK  
 TRY ME: -

9 FIELD #9 NAME: UNIT  
 9.1 UNIQUE TO RECORD? Y  
 9.2 TYPE: R  
 9.3 CARD TYPE: 1  
 9.4 COLUMNS: 34-37  
 9.5 CARD TYPE:  
 9.6 SYNTAX DEF: 999910000  
 TRY ME: -

Figure 7b. Research File Format Description

10 FIELD #10 NAME: JOB CODE  
10.1 UNIQUE TO RECORD? Y  
10.2 TYPE: R  
10.3 CARD TYPE: 38 1  
10.4 COLUMNS: 38-41  
10.5 CARD TYPE: -  
10.6 SYNTAX DEF: -C9.6 999910000  
TRY ME: -

11 FIELD #11 NAME: SKILL CODE  
11.1 UNIQUE TO RECORD? NO  
11.2 TYPE: R  
11.3 CARD TYPE: 3  
11.4 COLUMNS: 9-12  
11.5 SYNTAX DEF: -C9.6 999910000  
TRY ME: -

12 FIELD #12 NAME: SL\KILL NAME  
12.1 UNIQUE TO RECORD? N  
12.2 TYPE: R  
12.3 CARD TYPE: 3  
12.4 COLUMNS: 13-32  
12.5 SYNTAX DEF: <10X>[10X\\<10X>  
ERROR IN DEFINITION... PARENTHESES MUST COME IN PAIRS.  
12.5 SYNTAX DEF: <10X>[\\<10X>  
TRY ME: -

STORAGE AREA IS NEARLY FULL; PLEASE MAKE ALL NECESSARY CORRECTIONS  
TO THE ABOVE QUESTIONS.

13 FIELD #13 NAME: RECORD-IDENTIFYING FIELD:  
13.1 CARD TYPE: 1,2,3  
13.2 COLUMNS: 1-5  
13.3 SYNTAX DEF: XXXXX  
TRY ME: -

THE QUESTION NUMBERING WILL NOW START OVER.

1 GROUP NAME: SKILLS  
1.1 FIELDS IN THIS GROUP: 11,12  
2 GROUP NAME: -

END OF FILE FILE 10:25 AM 9/16/1965

-THANK YOU-

Figure 7c. Research File Formac Description

1 December 1965

3-157

TM-2624/100/00

1 DO YOU WANT LIST OF FILE NAMES? NO  
2 FILE NUMBER: 50 EMPLOYEE FILE 12:55 PM 9/16/1965  
2.1 CONFIDENTIAL CODE: SDC  
1 NUMBER OF TAPE UNIT? 1  
2 STARTING CARD NUMBER? 5  
3 UPDATE ONLY? N

-THANK YOU-

Figure 8. Research Card Input Program

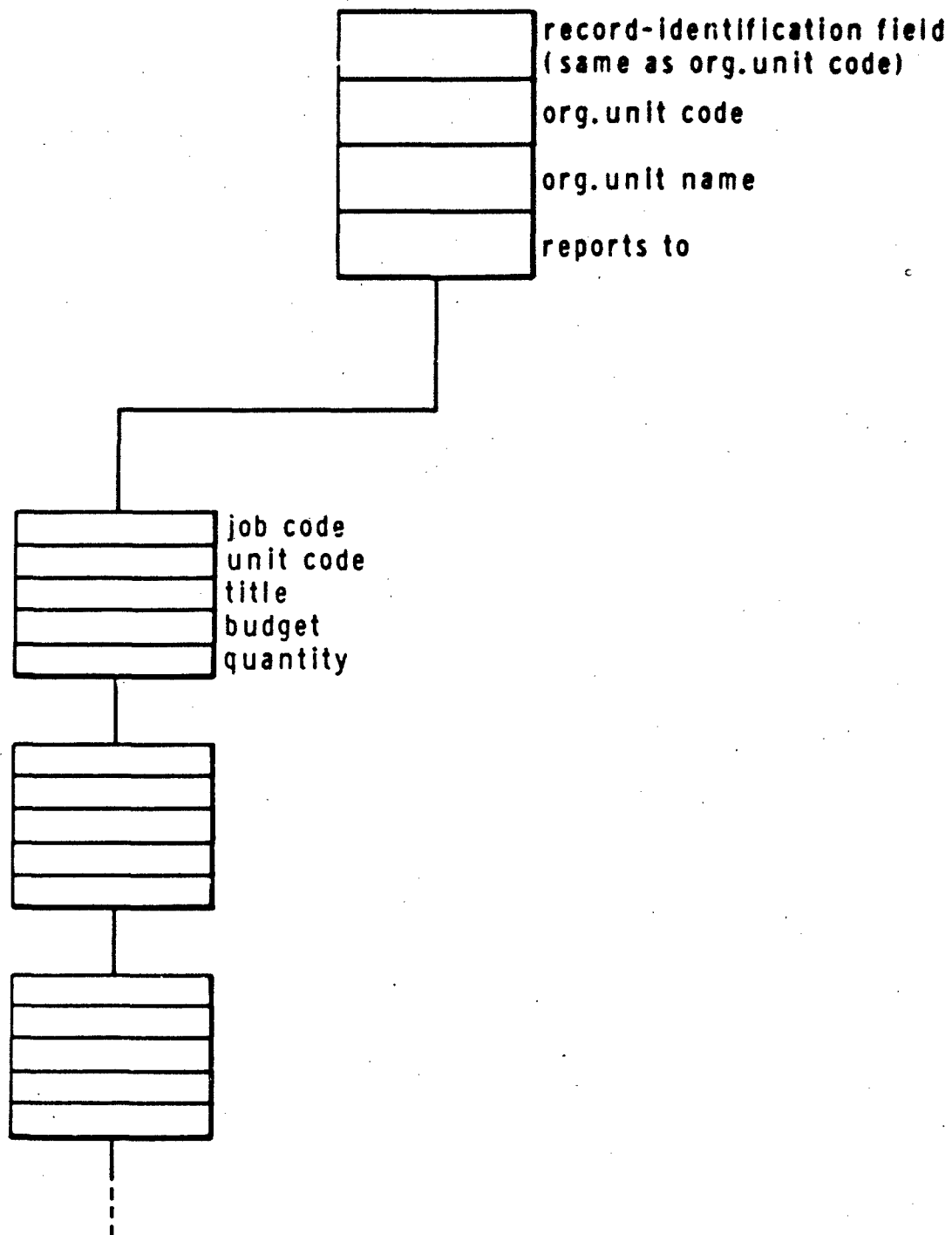


Figure 9. Structure of a Record in the Organization File

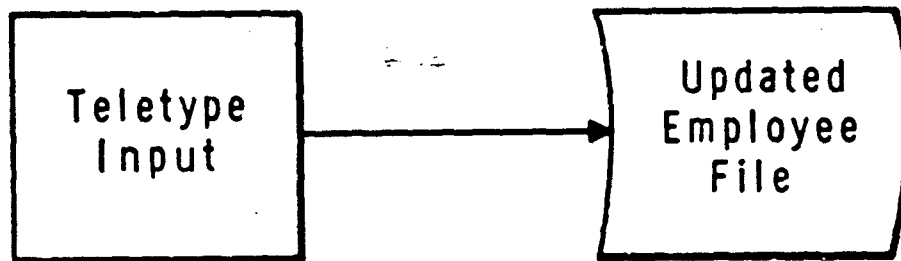


Figure 10. Updating Transaction

1 December 1965

3-160

TM-2624/100/00

```
1 FILE NAMES? NO
2 FILE: 50 EMPLOYEE FILE      2:04 PM 9/16/1965
  2.1 CONF. CODE: SDC
    0.1 REC-ID: 33144
1 NUMBER: -CO.1 33144
2 DATE OF TERMINATION: 09/16/1965
3 NAME: QUINN, S. M.
4 SALARY: 005500
5 SEX: F
6 BIRTHDATE: 111546
7 LEVEL: EMPL
8 POSITION TITLE: FILE CLERK
9 UNIT: 2123
10 JOB CODE: 5520
11 SKILL CODE:
   11.1 5520
   11.2 END
12 SKILL NAME:
   12.1 FILE CLERK
   12.2 END
  0.1 REC-ID: 91152
1 NUMBER: -CO.1 91152
2 DATE OF TERMINATION:
3 NAME: GARBER, B. E.
4 SALARY: 0008500 FIX
4 SALARY: 008500
5 SEX: M
6 BIRTHDATE: 070734
7 LEVEL: EMPL
8 POSITION TITLE: DRAFTSMAN
9 UNIT: 2111
10 JOB CODE: 1330
11 SKILL CODE:
   11.1 11\330
   11.2 END
12 SKILL NAME:
   12.1 DRAFTSMAN
   12.2 END
  0.1 REC-ID: 85657
1 NUMBER: -CO.1 85657
2 DATE OF TERMINATION:
3 NAME: LEE, R.E.
4 SALARY: 007200
5 SEX: M
6 BIRTHDATE: 031126
7 LEVEL: EMPL
8 POSITION TITLE: MACHINE OPR
9 UNIT: 2311
10 JOB CODE: 3340
11 SKILL CODE:
   11.1 3340
   11.2 3355
   11.3 3570
   11.4 END
12 SKILL NAME:
   12.1 MACHINE OPR
   12.2 PLATING OPR
   12.3 DISPATCH ASST
   12.4 END
  0.1 REC-ID:  
```

-THANK YOU-

Figure 11. Research Teletype Input Program



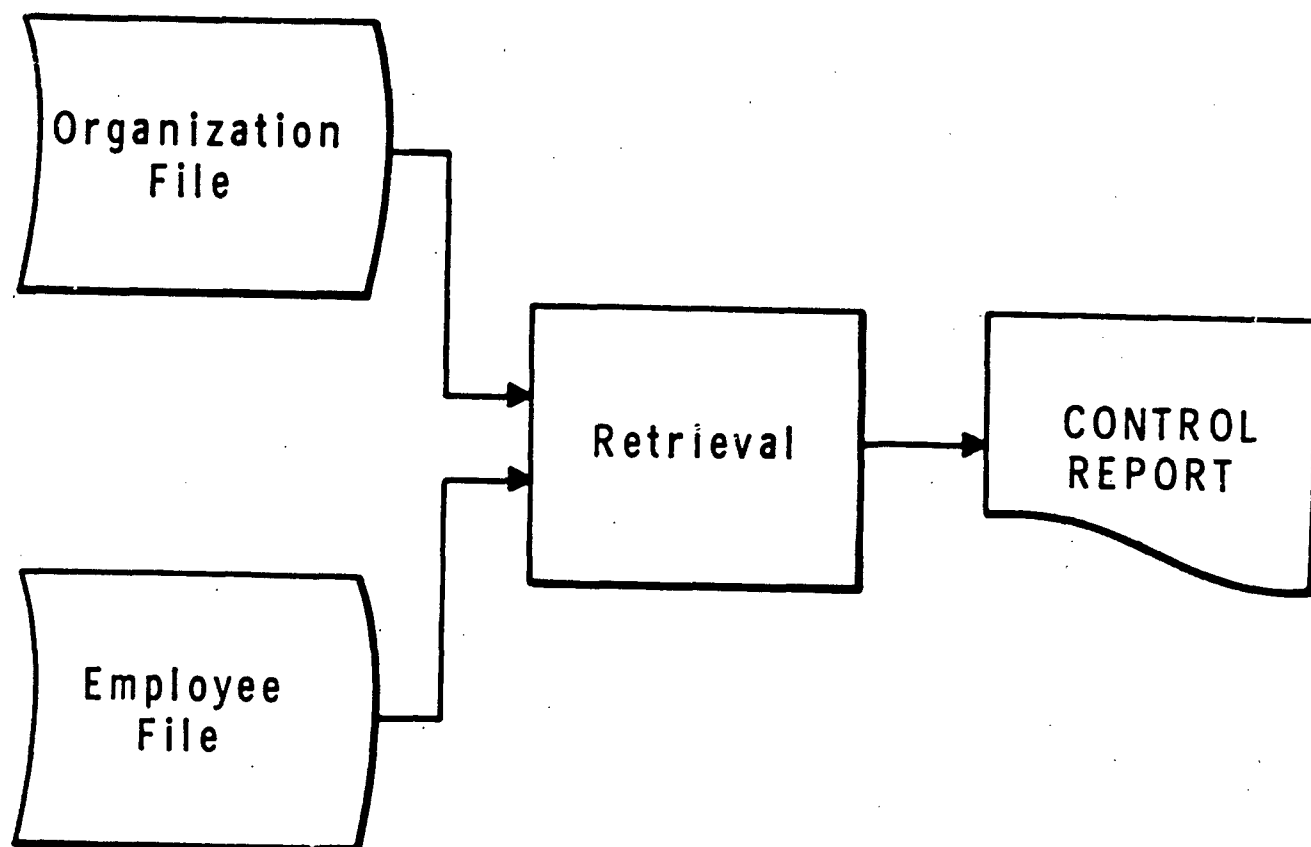


Figure 12. Control Report

1 December 1965

3-162

TM-2624/100/00

SEARCH PROGRAM  
11:55 AM 9/16/1965

```
1  FILE  51
2  DICT  N
3  NEW FIELDS
   3.1  FREQ ORG UNIT CODE:REDORDS
   3.2  -
4  DESCRIPTORS
   4.1
5  LIMITS
   5.1  POP      -
   5.2  MAX RECS -
   5.3  MAX POP  -
   5.4  MAX SAM  -
6  PRINT
   6.1  -
7  SUM
   7.1  FIELD  REDORDS:ORG
   7.2  FIELD  -
8  ROWS
   8.1  R1 SAMPLE
9  TITLE  NUMBER OF ORGANIZATION UNITS
10 OK  Y
```

11:55 AM 9/16/1965  
NUMBER OF ORGANIZATION UNITS

11:56 AM 9/16/1965  
DATA MATRICES ON FILE

ORG MATRIX

29.000

MATRIX FILE NO. = 10072

Figure 13. Search Program

1 December 1965

3-163

TM-2624/100/00

SEARCH PROGRAM  
3:29 PM 9/16/1965

```
1 FILE 50
2 DICT N
3 NEW FIELDS
  3.1 FREQ NUMBER:COUNT
  3.2 _
4 DESCRIPTORS
  4.1 NUMBER>00000:ALL
  4.2 _
5 LIMITS
  5.1 POP ALL
  5.2 MAX RECS _
  5.3 MAX POP _
  5.4 MAX SAM _
6 PRINT
  6.1 _
7 SUM
  7.1 FIELD ALL:EMP FIX
  7.1 FIELD COUNT:EMP
  7.2 FIELD SALARY:SAL
  7.3 FIELD _
8 ROWS
  8.1 R1 SAMPLE ALL
  8.2 R1 LABEL NUMBER OF PERSONS EMPLOYED
  8.3 R2 SAMPLE _
9 COLS
  9.1 C1 SAMPLE ALL
  9.2 C1 LABEL _
  9.3 C2 SAMPLE _
10 TITLE -8.2
  8.2 R1 LABEL NUMBER OF PERSONS EMPLOYED
10 TITLE NUMBER OF PERSONS EMPLOYED AND TOTAL ANNUAL SALARIES
NUMBER OF PERSONS EMPLOYED AND TOTAL ACTUAL ANNUAL SALARIES
11 OK Y
```

3:35 PM 9/16/1965  
NUMBER OF PERSONS EMPLOYED AND TOTAL ACTUAL ANNUAL SALARIES  
EMP MATRIX

104.000

SAL MATRIX

967850.000

MATRIX FILE NO. = 10323

Figure 14. Search Program

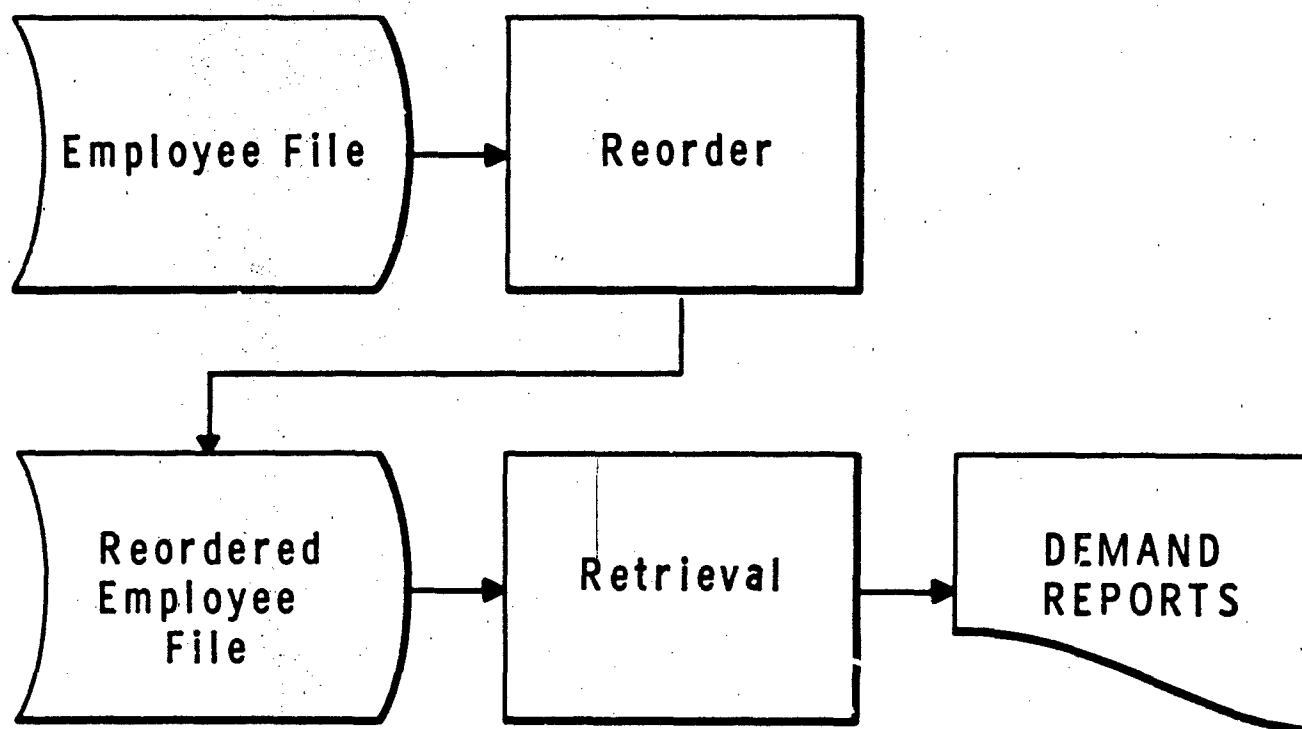


Figure 15. Demand Reports

1 December 1965

3-165

TM-2624/100/00

"G"  
SHUFFLE PROGRAM  
225 PM 9/16/1965  
1 DO YOU WANT LIST OF FILE NAMES? NO  
2 FILE NUMBER: 50 EMPLOYEE FILE 2:05 PM 9/16/1965  
2.1 CONFIDENTIAL CODE: SDC  
3 DESIRED ORDERING OF FILE: NAME  
NAME  
55 EMPLOYEE FILE NAME 2:05 PM 9/16/1965  
  
-THANK YOU-

Figure 16. Shuffle Program

1 December 1965

3-166

TM-2624/100/00

SEARCH PROGRAM

4:09 PM 9/16/1965

```
1  FILE  55
2  DICT  N
3  NEW FIELDS
   3.1  FREQ SKILL CODE:TOTAL NUMBER OF SKILLS
   3.2
4  DESCRIPTORS
   4.1  SKILL CODE=3340:D1
   4.2  TOTAL NUMBER OF SKILLS>2 AND SEX=M AND LEVEL NOT=HEAD:D2
   4.3  BIRTHDATE>12/31/1916 AND BIRTHDATE<01/01/1935:D3
   4.4  SEX=M AND LEVEL NOT=HEAD:D4
   4.5  SKILL CODE=1110:D5
   4.6  SKILL CODE>1129 AND SKILL CODE<1140:D6
   4.7
5  LIMITS
   5.1  POP D4 AND (D5 OR D6)
   5.2  MAX RECS  _
   5.3  MAX POP   _
   5.4  MAX SAM   _
6  PRINT
   6.1  NAME
   6.2  NUMBER
   6.3  UNIT
   6.4  SKILL CODE
   6.5  SKILL NAME
   6.6  _
       6.6.1 PRINT IND _
7  SUM
   7.1  FIELD  _
8  ROWS
   8.1  R1 SAMPLE _
9  TITLE DEMAND REPORT #10Y
10 OK Y
```

Figure 17a. Search Program

1 December 1965

3-167

TM-2624/100/00

3:46 PM 9/16/1965  
DEMAND REPORT #1

NAME	BOYD, W. V.
NUMBER	15052
UNIT	2135
SKILL CODE	SKILL NAME
1351	MECH TECHN
3340	MACHINE OPR
3370	PAINTING OPR
7320	MAINT TECH E
NAME	COATES, C. L.
NUMBER	81130
UNIT	2313
SKILL CODE	SKILL NAME
3340	MACHINE OPR
3345	MILLING MACH OPR
3360	HEAT TREAT OPR
NAME	FLETCHER, M. W.
NUMBER	21475
UNIT	2133
SKILL CODE	SKILL NAME
1365	TOOL DESIGNER
3125	TOOL MAKER
3340	MACHINE OPR
7320	MAINT TECH E
NAME	GORTON, R. A.
NUMBER	17590
UNIT	2313
SKILL CODE	SKILL NAME

Continued ....

Figure 17b. Demand Report

1 December 1965

3-168

TM-2624/100/00

4:04 PM 9/16/1965  
DEMAND REPORT #10Y

NAME	APGAR, A. J.
NUMBER	82802
UNIT	2115
SKILL CODE	SKILL NAME
1130	ELEC ENGR

NAME	ARNETTE, L. J.
NUMBER	37113
UNIT	2115
SKILL CODE	SKILL NAME
1130	ELEC ENGR

NAME	ETTINGER, G. J.
NUMBER	13581
UNIT	2113
SKILL CODE	SKILL NAME
1130	ELEC ENGR

Continued.....

NAME	STADERMAN, P. K.
NUMBER	80823
UNIT	2122
SKILL CODE	SKILL NAME
1130	ELEC ENGR

RECORDS SEARCHED =	104
RECORDS SELECTED =	10
RECORDS INDETERM =	0
SAMPLE SELECTED =	0
SAMPLE INDETERM =	0

Figure 18. Demand Report



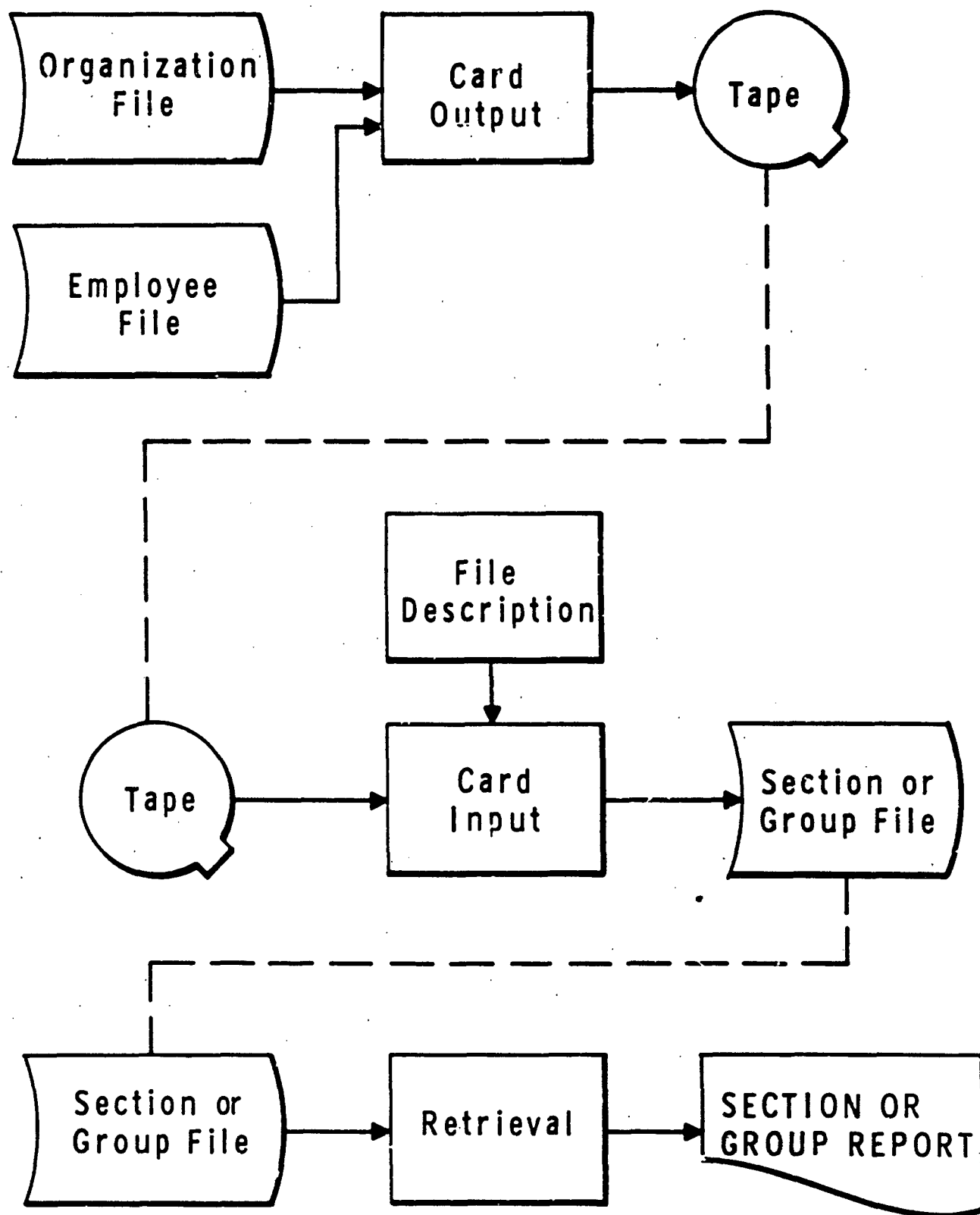


Figure 19. Periodic Reports

1 December 1965

3-170

TM-2624/100/00

100"G"

POUT PROGRAM  
10:48 AM 9/17/1965

1 LIST OF FILE NAMES? N  
2 FILE. 51 ORGANIZATION FILE 11:43 AM 9/16/1965  
2.1 CONFIDENTIAL CODE: BBN  
3 TAPE UNIT 0  
FIELD, FROM COL., THROUGH COL.  
4.1 LOAD INFO. ORG UNIT CODE,1,4  
4.2 LOAD INFO. ORG UNIT NAME,5,24  
4.3 LOAD INFO. REPORTS TO,25,28  
4.4 LOAD INFO. JOB CODE,29,32  
4.5 LOAD INFO. UNIT CODE,33,36  
4.6 LOAD INFO. TITLE,37,56  
4.7 LOAD INFO. BUDGET,59,64  
4.8 LOAD INFO. QUANTITY,65,66  
4.9 LOAD INFO. -  
OUR WORK IS DONE.

-THANK YOU-

POUT PROGRAM  
11:20 AM 9/17/1965

1 LIST OF FILE NAMES? N  
2 FILE. 50 EMPLOYEE FILE 2:05 PM 9/16/1965  
2.1 CONFIDENTIAL CODE: SDC  
3 TAPE UNIT 0  
FIELD, FROM COL., THROUGH COL.  
4.1 LOAD INFO. UNIT,1,4  
4.2 LOAD INFO. JOB CODE,29,32  
4.3 LOAD INFO. POSITION TITLE,37,56  
4.4 LOAD INFO. SALARY,59,64  
4.5 LOAD INFO. -  
OUR WORK IS DONE.

-THANK YOU-

Figure 20. POUT Program

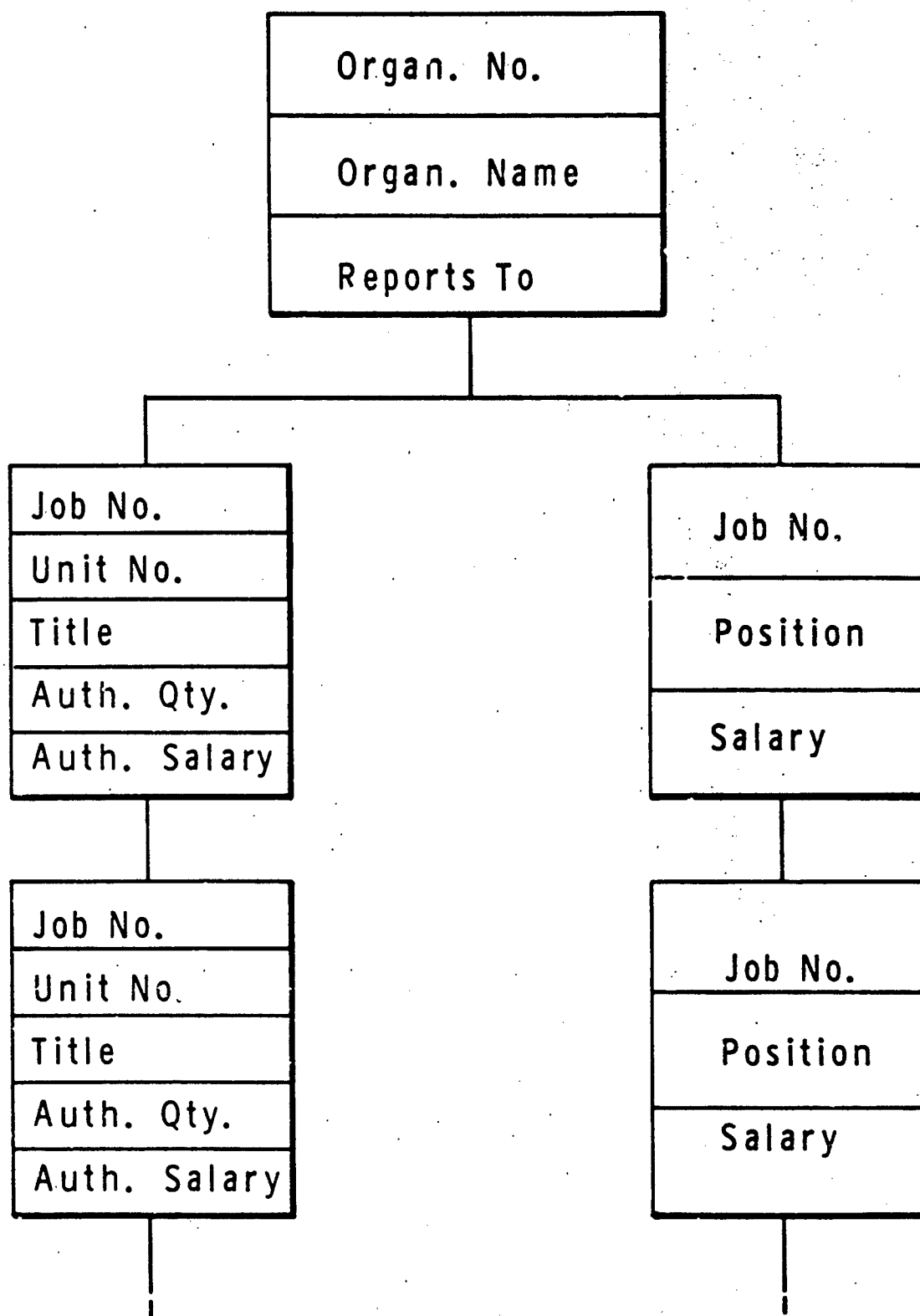


Figure 21. Section or Group File

0.1 FILE TITLE: SECTION FILE  
0.2 CONFIDENTIAL CODE: -  
0.3 NO. OF CARD-TYPES: 3  
    0.3.1 DEFINING COLUMNS FOR TYPE #1: 80  
        0.3.1.1 SYNTAX DEF: "1";"2"  
        0.3.1.2 NO. OF CONTINUATION CARDS: 1  
        0.3.1.3 IDENTIFYING COLUMNS: 1-5  
    0.3.2 DEFINING COLUMNS FOR TYPE #2: 80  
        0.3.2.1 SYNTAX DEF: "3"  
        0.3.2.2 NO. OF CONTINUATION CARDS: 0  
    0.3.3 TYPE #3 IS ASSUMED TO BE ALL THE OTHERS.  
        0.3.3.1 NO. OF CONTINUATION CARDS: 0

1 FIELD #1 NAME: ORG UNIT CODE  
    1.1 UNIQUE TO RECORD? Y  
    1.2 TYPE: R  
    1.3 CARD TYPE: 1  
    1.4 CARD NO: 1  
    1.5 COLUMNS: 34-37  
    1.6 CARD TYPE: 3  
    1.7 COLUMNS: 1-4  
    1.8 CARD TYPE: -  
    1.9 SYNTAX DEF: 9999I" "

2 FIELD #2 NAME: ORG UNIT NAME  
    2.1 UNIQUE TO RECORD? Y  
    2.2 TYPE: R  
    2.3 CARD TYPE: 3  
    2.4 COLUMNS: 5-24  
    2.5 CARD TYPE: -  
    2.6 SYNTAX DEF: XXXXXXXXXXXXXXXXXXXXX=<20X>

3 FIELD #3 NAME: REPORTS TO  
    3.1 UNIQUE TO RECORD? Y  
    3.2 TYPE: R  
    3.3 CARD TYPE: 3  
    3.4 COLUMNS: 25-28  
    3.5 CARD TYPE: -  
    3.6 SYNTAX DEF: 9999;####

4 FIELD #4 NAME: AUTHORIZED COMPLEMENT  
    4.1 UNIQUE TO RECORD? NO  
    4.2 TYPE: R  
    4.3 CARD TYPE: 3  
    4.4 COLUMNS: 29-56  
    4.5 SYNTAX DEF: [[####];"..";XXXX][[####];"..";XXXX]I" "<20X>

Figure 22a. Research File Format Description

- 5 FIELD #5 NAME: QUANTITY  
5.1 UNIQUE TO RECORD? NO  
5.2 TYPE: I  
5.3 CARD TYPE: 3  
5.4 COLUMNS: 57-58  
5.5 SYNTAX DEF: 99;##
- 6 FIELD #6 NAME: SALARY  
6.1 UNIQUE TO RECORD? NO  
6.2 TYPE: I  
6.3 CARD TYPE: 3  
6.4 COLUMNS: 59-64  
6.5 SYNTAX DEF: 999999;#####
- 7 FIELD #7 NAME: ACTUAL COMPLEMENT  
7.1 UNIQUE TO RECORD? N  
7.2 TYPE: R  
7.3 CARD TYPE: 1  
7.4 CARD NO: 1  
7.5 COLUMNS: 38-75  
7.6 SYNTAX DEF: XXXX[XXXX];".. "<20X>XXXXXXXXXX
- 8 FIELD #8 NAME: SALARY  
8.1 UNIQUE TO RECORD? NO  
8.2 TYPE: I  
8.3 CARD TYPE: 1  
8.4 CARD NO: 2  
8.5 COLUMNS: 16-21  
8.6 SYNTAX DEF: 999999;#####
- 9 FIELD #9 NAME: RECORD-IDENTIFYING FIELD:  
9.1 CARD TYPE: 1,2  
9.2 COLUMNS: 34-37  
9.3 CARD TYPE: 3  
9.4 COLUMNS: 1-4  
9.5 SYNTAX DEF: XXXX
- 10 GROUP NAME: AUTH GROUP  
10.1 FIELDS IN THIS GROUP: 4,5,6
- 11 GROUP NAME: ACTUAL GROUP  
11.1 FIELDS IN THIS GROUP: 7,8
- 12 GROUP NAME: —

59 SECTION FILE 1:56 AM 9/17/1965

-THANK YOU-

1 December 1965

3-174

TM-2624/100/00

2:00 AM 9/17/1965  
CALL RESCDI

RESEARCH CARD INPUT PROGRAM

1 DO YOU WANT LIST OF FILE NAMES? N  
2 FILE NUMBER: 59 SECTION FILE 1:56 AM 9/17/1965  
1 NUMBER OF TAPE UNIT? 0  
2 STARTING CARD NUMBER? 1  
3 UPDATE ONLY? NO

-THANK YOU-

Figure 23. Research Card Input Program

1 December 1965

3-175

TM-2624/100/00

SEARCH PROGRAM  
2:32 AM 9/17/1965

```
1  FILE  59
2  DICT  N
3  NEW FIELDS
   3.1  SUM QUANTITY:TOTAL AUTH. QUANTITY
   3.2  SUM SALARY:TOTAL AUTH. SALARY
   3.3  FREQ SALARY:TOTAL ACTUAL QUANTITY
   3.4  SUM SALARY:TOTAL ACTUAL SALARY
   3.5  TOTAL ACTUAL SALARY-TOTAL AUTH. SALARY:DEVIATION FROM BUDGET
   3.6
4  DESCRIPTORS
   4.1  ORG UNIT CODE CONTAINS "O ":NS
   4.2
5  LIMITS
   5.1  POP  NOT NS
   5.2  MAX RECS  -
   5.3  MAX FOP   -
   5.4  MAX SAM   -
6  PRINT
   6.1  ORG UNIT CODE
   6.2  REPORTS TO
   6.3  ORG UNIT NAME
   6.4  AUTHORIZED COMPLEMENT
   6.5  SALARY
   6.6  QUANTITY
   6.7  TOTAL AUTH. SALARY
   6.8  TOTAL AUTH. QUANTITY
   6.9  ACTUAL COMPLEMENT
   6.10 SALARY
   6.11 TOTAL ACTUAL SALARY
   6.12 TOTAL ACTUAL QUANTITY
   6.13 DEVIATION FROM BUDGET
   6.14
       6.14.1 PRINT IND Y
7  SUM
   7.1 FIELD  -
8  ROWS
   8.1 R1 SAMPLE  -
9  TITLE PERIODIC SECTION REPORT
10 OK Y
```

Figure 24a. Search Program

1 December 1965

3-176

TM-2624/100/00

2:33 AM 9/17/1965  
PERIODIC SECTION REPORT

ORG UNIT CODE 2122  
REPORTS TO 2120  
ORG UNIT NAME SYSTEM STDS. SECTION

AUTHORIZED COMPLEMENT	SALARY	QUANTITY
1120.. CHIEF	11500	1
1120.. MECH ENGR	9500	1
1130.. ELEC ENGR	9700	1
1330.. DRAFTSMAN	8500	1
TOTAL AUTH. SALARY	39200.0000000	
TOTAL AUTH. QUANTITY	4.0000000	

ACTUAL COMPLEMENT	SALARY
1120.. CHIEF, SYSTEM STDS. SECTION	11500
1130.. ELECENGR	9700
1120.. MECH ENGR	9500
TOTAL ACTUAL SALARY	30700.0000000
TOTAL ACTUAL QUANTITY	3.0000000
DEVIATION FROM BUDGET	-8500.0000000

ORG UNIT CODE 2123  
REPORTS TO 2120  
ORG UNIT NAME COMPONENT STDS SECT

AUTHORIZED COMPLEMENT	SALARY	QUANTITY
1120.. CHIEF	11000	1
1120.. MECH ENGR	9500	1
1130.. ELEC ENGR	10000	1
1330.. DRAFTSMAN	8500	1
5520.. FILE CLERK	5500	1

TOTAL AUTH. SALARY	44500.0000000
TOTAL AUTH. QUANTITY	5.0000000

ACTUAL COMPLEMENT	SALARY
1330.. DRAFTSMAN	8500
1130.. ELEC ENGR	10000
5520.. FILE CLERK	5500
1120.. CHIEF, COMPONE NT STDS. SECTION	11000
1120.. MECH ENGR	9500
TOTAL ACTUAL SALARY	44500.0000000
TOTAL ACTUAL QUANTITY	5.0000000
DEVIATION FROM BUDGET	0.0000000

Figure 24b. Periodic Section Report



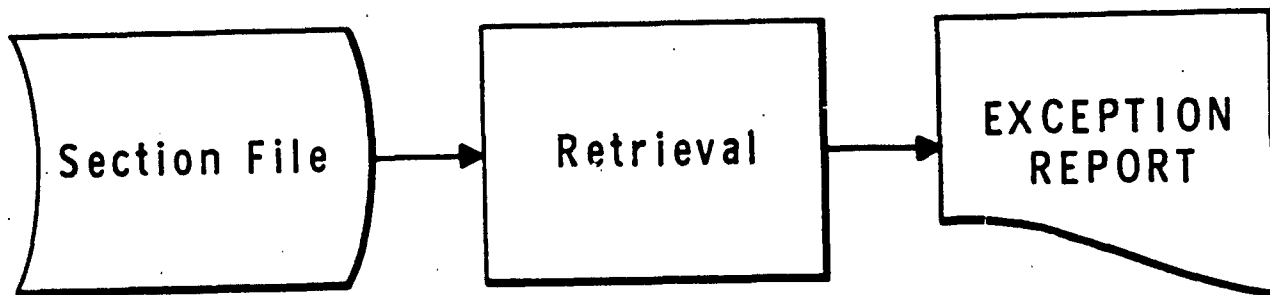


Figure 25. Exception Report

SEARCH PROGRAM  
4:02 AM 9/17/1965

```
1 FILE 59
2 DICT N
3 NEW FIELDS
  3.1 SUM QUANTITY:TOTAL AUTH. QUANTITY
  3.2 SUM SALARY:TOTAL AUTH. SALARY
  3.3 FREQ SALARY:TOTAL ACTUAL QUANTITY
  3.4 SUM SALARY:TOTAL ACTUAL SALARY
  3.5 TOTAL ACTUAL SALARY-TOTAL AUTH. SALARY:DEVIATION FROM BUDGET
  3.6 TOTAL AUTH. QUANTITY - TOTAL ACTUAL QUANTITY:QDIF
  3.7
4 DESCRIPTORS
  4.1 ORG UNIT CODE CONTAINS "0 ":NS
  4.2 QDIF=0 OR QDIF<0:D5
  4.3 DEVIATION FROM BUDGET<-799 OR DEVIATION FROM BUDGET>799:D6
  4.4
5 LIMITS
  5.1 POP NOT NS AND D5 AND D6
  5.2 MAX RECS -
  5.3 MAX POP -
  5.4 MAX SAM -
6 PRINT
  6.1 ORG UNIT CODE
  6.2 REPORTS TO
  6.3 ORG UNIT NAME
  6.4 AUTHORIZED COMPLEMENT
  6.5 SALARY
  6.6 QUANTITY
  6.7 TOTAL AUTH. SALARY
  6.8 TOTAL AUTH. QUANTITY
  6.9 ACTUAL COMPLEMENT
  6.10 SALARY
  6.11 TOTAL ACTUAL SALARY
  6.12 TOTAL ACTUAL QUANTITY
  6.13 DEVIATION FROM BUDGET
  6.14
    6.14.1 PRINT IND Y
7 SUM
  7.1 FIELD -
8 ROWS
  8.1 R1 SAMPLE -
9 TITLE EXCEPTION REPORT
10 OK Y
```

Figure 26a. Search Program

1 December 1965

3-179

TM-2624/100/00

4:04 AM 9/17/1965  
EXCEPTION REPORT

ORG UNIT CODE 2113  
REPORTS TO 2110  
ORG UNIT NAME ADV. SYSTEMS SECTION

AUTHORIZED COMPLEMENT	SALARY	QUANTITY
1110.. CHIEF	13000	1
1110.. SYSTEMS ENGR	12000	1
1120.. MECH ENGR	12000	1
1130.. ELEC ENGR	12000	1
TOTAL AUTH. SALARY	49000.0000000	
TOTAL AUTH. QUANTITY	4.0000000	

ACTUAL COMPLEMENT	SALARY
1130.. ELEC ENGR	11000
1110.. SYSTEMS ENGR	12000
1120.. MECH ENGR	12000
1110.. CHIEF, ADV. SYSTEMS SECTION	13000
TOTAL ACTUAL SALARY	48000.0000000
TOTAL ACTUAL QUANTITY	4.0000000
DEVIATION FROM BUDGET	-1000.0000000

ORG UNIT CODE 2115  
REPORTS TO 2110  
ORG UNIT NAME PROD. SPEC. SECTION

AUTHORIZED COMPLEMENT	SALARY	QUANTITY
1110.. CHIEF	12000	1
1120.. MECH ENGR	11000	1
1130.. ELEC ENGR	11000	1
1330.. DRAFTSMAN	8000	1
TOTAL AUTH. SALARY	42000.0000000	
TOTAL AUTH. QUANTITY	4.0000000	

ACTUAL COMPLEMENT	SALARY
1110.. CHIEF, PROD. SPEC. SECTION	12000
1120.. MECH ENGR	11000
1130.. ELEC ENGR	11000

Figure 26b. Exception Report

1 December 1965

3-180

TM-2624/100/00

1330.. DRAFTSMAN	8000
1130.. ELEC ENGR	10500
TOTAL ACTUAL SALARY	52500.0000000
TOTAL ACTUAL QUANTITY	5.0000000
DEVIATION FROM BUDGET	10500.0000000

ORG UNIT CODE	2311
REPORTS TO	2310
ORG UNIT NAME	FAB. SECTION A

AUTHORIZED COMPLEMENT	SALARY	QUANTITY
3340.. CHIEF	10000	1
3340.. MACHINE OPR	40000	5

TOTAL AUTH. SALARY	50000.0000000
TOTAL AUTH. QUANTITY	6.0000000

ACTUAL COMPLEMENT	SALARY
3340.. CHIEF, FAB. SECTION A	10000
3340.. MACHINE OPR	6500
3340.. MACHINE OPR	8000
3340.. MACHINE OPR	8500
3340.. MACHINE OPR	9000
3340.. MACHINE OPR	7200

TOTAL ACTUAL SALARY	49200.0000000
TOTAL ACTUAL QUANTITY	6.0000000
DEVIATION FROM BUDGET	-800.0000000

RECORDS SEARCHED =	30
RECORDS SELECTED =	3
RECORDS INDETERM =	0
SAMPLE SELECTED =	0
SAMPLE INDETERM =	0

Figure 26c. Exception Report (Cont'd)

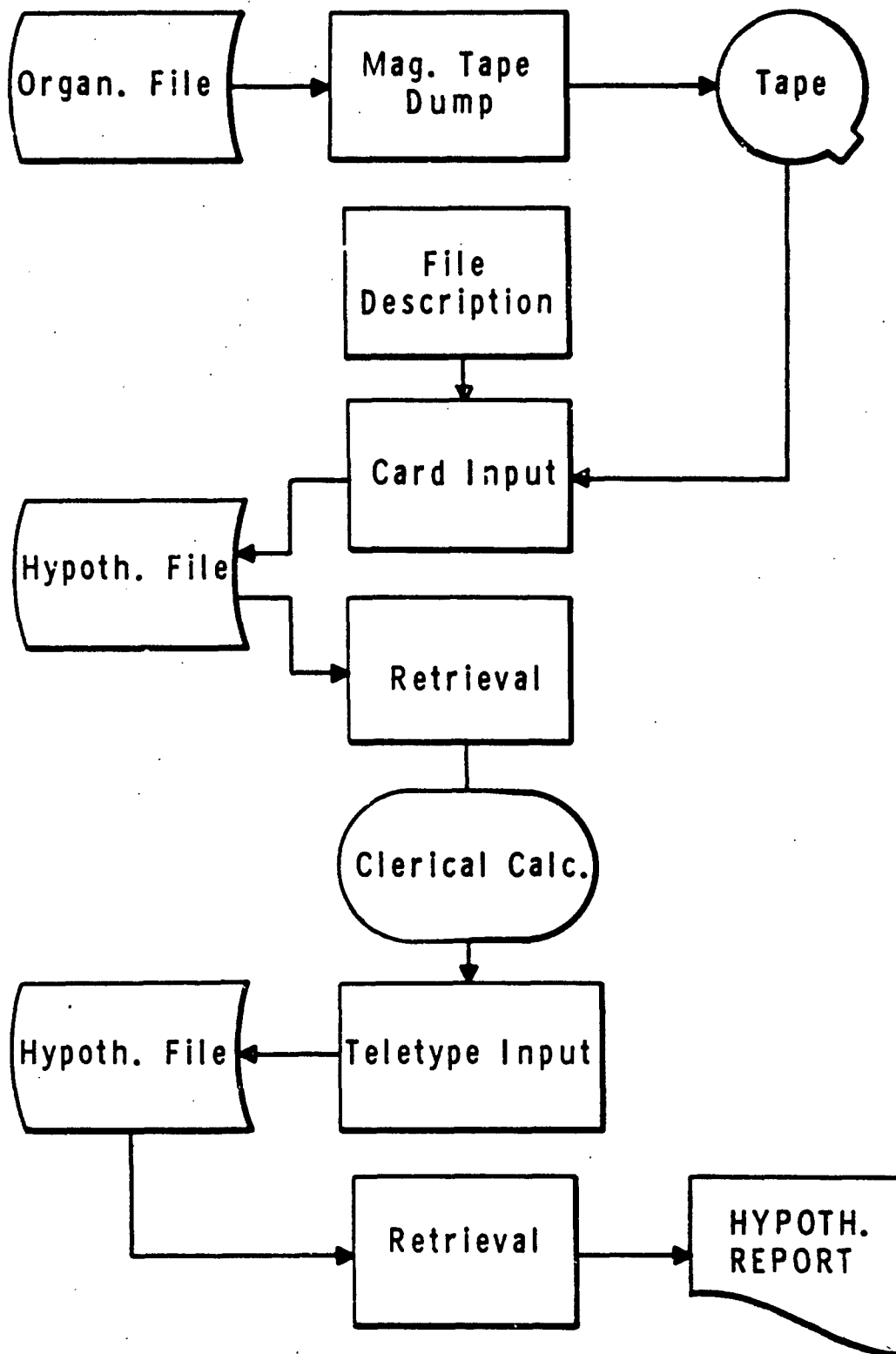


Figure 27. Hypothesis Report

QUESTION:

In the half day of training, does the hospital staff learn much sophistication?

ANSWER:

To date, the system has been used for very small things.

QUESTION:

Can you get high-volume output on a line printer, or does everything come out on the teletype?

ANSWER:

We are getting a high-speed printer, but we haven't found the need for it yet. Because of the time-sharing principle, we just let it run during the night and it comes out in the morning.

QUESTION:

How do the nurses, doctors and all adapt to the names, the terminology, and the discipline of the system?

ANSWER:

Actually it's much better than we thought. There is a lot of human engineering I didn't show. There are a lot of other modes of asking a question when you don't know the answer. For instance, you can ask "how," and the system should type out a long type script of "how." This is done in the way that the SDC time-sharing system gives an estimation of how to do things.

QUESTION:

Have there been any studies of an easier terminology for the nurses to understand?

ANSWER:

What you saw wouldn't be used by actual nurses on the floor because they are doing routine things. In operation, we have actual programs specific to giving a patient a drug, and the terminology is catered exactly to the way the nurses are accustomed to it.

QUESTION:

Are there any other general types of format difficulties?

ANSWER:

No, one of the significant factors of this system was the creation of sub-totals. In medical research they aren't so interested in subtotals. They are more interested in high-speed analyses. And they don't actually get these long printouts--they condense it down into some more readable form. Because of the give and take of time-sharing, most people don't want long printouts. They usually only get a couple of records.

QUESTION:

Is the operation of the system reminiscent of Project MAC?

ANSWER:

Yes, it is. The chief difference is that Project MAC is a generalized time-sharing system, and our hospital system is specific to a particular type of time-sharing.

QUESTION:

How long did it take to program the problem?

ANSWER:

I was given the problem last Tuesday, started it on Thursday, and finished it on Friday, taking advantage of the time-sharing system.

## BEST SYSTEM

Presented by: J. R. Ziegler  
Manager of Programming Services  
National Cash Register Company

The specific project at hand was to solve a given, well-defined problem using BEST (Business EDP Systems Technique).

The BEST technique started out in a very similar, problem-solving situation about five years ago in the Data Processing Centers of the National Cash Register Company. It was originally implemented on the NCR 304 and was subsequently enlarged on the basis of experience and adapted to the NCR 315. It has been operational on the 315 for more than two years and has been applied by hundreds of users.

BEST is not designed to work in on-line or time-sharing systems at the present time. Its purpose is to get a job onto a computer, mainly ours, as fast as possible and as efficiently as possible. This is done through a series of program generators similar to sort generators. These cover some 45 functions common to the processing of business data processing applications.

The first requirement in defining the solution to the particular problem at hand was a decision on how to organize the files. Although this particular job constituted a specific set of requirements, the proviso was also included that we should realize there may be a larger file involved. There may also be other files, other fields, or other reports involved; and there may be other data manipulations required.

We approached the job in such a way that each of the reports or programs which we had to produce was independent of the other. This was because, at any time, our hypothetical boss might ask for a different set of reports from the same files. Or he might ask that our two files be combined with a third file, such as one covering physical location. In order to be able to meet such requirements, the decision was made to maintain the organization file and the personnel file, which contain separate types of information, as two independent files.

Admittedly, these two files talk to each other. But they do so in only one direction: The personnel file provides data on people, jobs or positions to the organization file. The first step, then, was to set up the personnel file by itself. Since most of the changes to the personnel file would come identified only by the person's number, the file was organized according to employee ID number. Fields therein are completely independent, since a field, if it exists, must be changeable. We set up this particular operation to change each field independently of each other field.



The organization file is, in essence, the same kind of a file. So, the flow charts and other documentation which will be illustrated for the personnel file could be duplicated to cover both requirements.

There will be five different types of documentation in the illustrations which show the solution to this problem:

1. There will be flow charts drawn on a functional basis. These charts were drawn at the planning stage of the problem and, interestingly enough, they also serve as documentation at the final level of the program. The flow charts that one writes at the beginning--at the thinking level in a BEST program--are the same flow charts that one implements when the job is actually running.

2. Data Record Layout forms will be illustrated. These are filled out in the initial, planning stages of a program and are retained as part of the final permanent documentation available for reference by anyone--either management or data processing personnel--working with the system at any time. For each BEST function, there is a Data Record Layout form which serves as a method of disciplining the formatting of information. These forms call for specific listings of the input data as the computer will see it and for the form the data will take in outputting information or in interfacing with the next BEST function.

3. Each function also requires the completion of a Parameter Sheet which describes exactly how the data is to be processed or manipulated in this segment of the program. Parameter Sheets contain a series of questions, written in English, which call for answers in terms of specific numbers, directions, or yes/no answers. By answering the questions, the programmer lets the system know which of the capabilities of the function he wishes to apply and exactly what operations are to be performed.

4. The illustrations will also include samples of coding printed out automatically under the control of the BEST generator.

5. Finally, there will be a number of reports printed by the BEST-generated program. These reports furnish the solution to the stated problem.

Figure 1 presents a Data Record Layout. There is one record of this type for each of the individuals on the personnel file. As you can see, the record contains employee number, employee name, use code, job code, level, salary, and all the rest of the information applicable to the personnel file. For this particular application, the employee name is maintained as a fixed-length field rather than as a variable-length field. Also, the number of skill codes is considered as a fixed number. This is because, under normal circumstances, even though you might have an indefinite number of secondary field codes, the number specified isn't at all unreasonable, and we have provided enough room for that

TM-26 24/100/00

and no more at the present time. If you have more, then you would extend the record. For example, you could have additional fields for such items as check number for last pay period or other kinds of information if you wished to incorporate these as part of your personnel file.

Figure 2 represents four different records in the organization file. This particular file has four levels. These are not strictly levels in the sense of a data hierarchy, but four different record codes. The space for Record Type (second from the left on the top line) indicates whether it is an organization unit name (top record) or a job name keyed for the manager only (second line). The third line down from the top represents the job name itself and the fourth represents the suborganization name.

In a job name layout, the second and third records would have in them the number of units and the salaries budgeted for this particular job. The two fields on the left represent either the organization unit and the "Reports To" or the organization unit and the job code.

If you have other levels of data, you would have other records. You can have as many of these as you want for any organizational level and for as many job codes as you have within any organization.

These two documents, then, represent the basic files under consideration.

The solution for the entire problem is flowcharted in Figure 3, except that we have not put the hypothesis section of the report on this particular flow chart. This chart calls for 13 separate computer runs. Allowing two more for the hypothesis reports, this makes a total of 15 computer runs if we include all programs, such as sorting, editing and reporting.

As laid out, once you have your Master File in existence, you could use any of the programs which produce any of the reports independently of any of the other ones. The only exception, I believe, is the Exception Report, which does utilize the outputs of the Section and Group Reports.

Now we will break this down into pieces as it was actually solved. As we saw it, the first job (Figure 4) was to create and update the Master File. This illustration represents the use of three different programs to generate or update this file. In this particular case, as in almost all cases of this kind, both updating and initial generation are done with the same, identical set of programs. This is because the initial generation of a Master File is really nothing more than the updating of a nonexistent file.





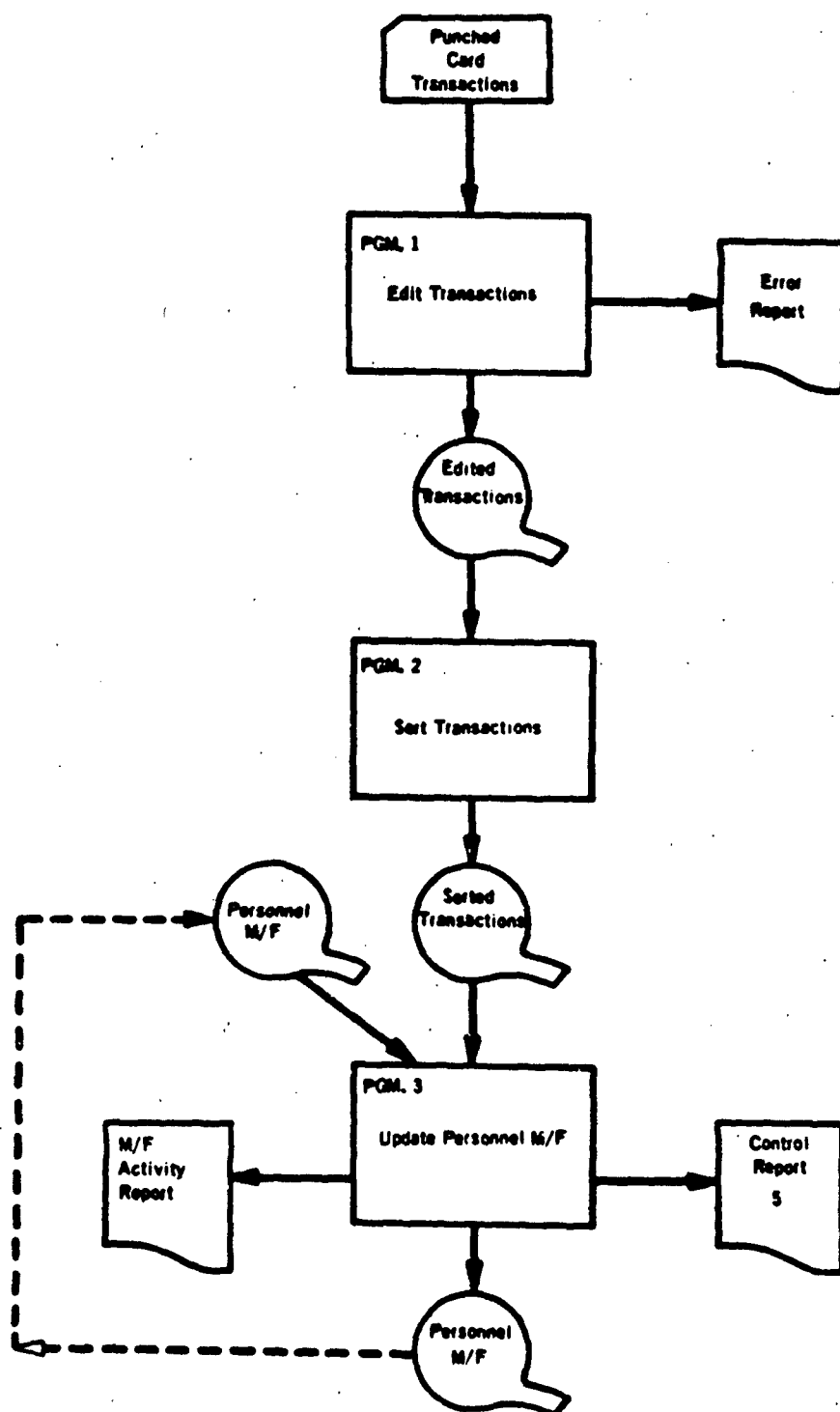


Figure 4. Updating Personnel Master File

The next illustration (Figure 5) is a flow chart of the entire program needed to read in and validate the punched card transactions which were used to update the Master File, as well as to create an error report and put the transaction data out for sorting purposes. This operation completely validates any type of transaction you may have. For example, knowing keypunch operators, you will sometimes get alphabetic entries in the middle of fields which call for numeric such as job code. This program incorporates complete validations to avoid entering any data of this type.

Each of the rectangular boxes on the flow chart represents a separate function. In this particular memory load program we have 17 functions. Each one of these is a common, basic function. For example, if we say READ PUNCHED CARDS every time the flow chart shows this designation, we know exactly what is going on. Now the parameters for reading cards may be different in the sense that you may use different columns or get them from a different card reader, and so on. However, from a communications point of view, the process is unequivocal. This is as true for the person who wrote the program and wants to alter it six months later, as it is for persons who succeed him on the job, for supervisors or for subordinates. The initial documentation of BEST is permanent.

The next flow chart, shown in Figure 6, represents the entire Master File updating memory load. It also covers the reporting of Master File maintenance transactions as well as the printing of the control report which was requested. This particular segment of the program may have 10 or 15 options, any of which can be provided for by filling out a Parameter Sheet. As a Master File operating system, then, this represents the entire memory load. It also accumulates the information for the control report as required.

The Master File operating function has tied with it several other functions which are called Master File Replacement Functions. There is a separate function of this type for each different kind of transaction. Figure 7 is an example showing several types of transactions necessary to maintain this particular Master File. If you had more transactions--in other words, more fields to edit--then you would have different Master File Replacement functions for each of the different transaction codes.

Figure 8 is an example of one of the pages of Parameter Sheets. This happens to be the Compute parameter. In calling this particular Parameter Sheet into play, you have a sequence number at the top which represents the actual number of the function in the flow charts. Entries on the Parameter Sheet denote the input area for this function--where the data is if the function is going to find it. Also shown is the output area, designating where the function is going to put the data after it has manipulated it. Down at the bottom, you fill out the different things that you want this particular function to do.







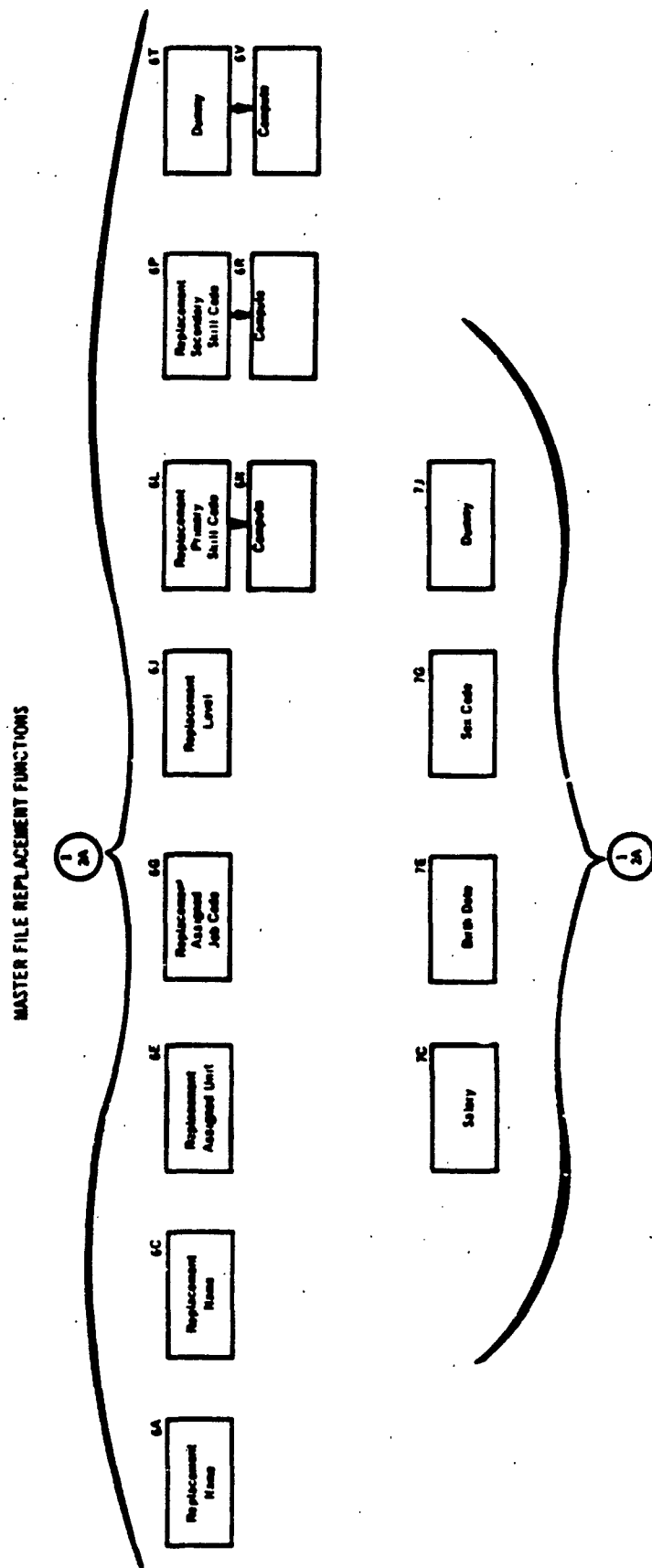


Figure 7. Master File Replacement Functions

1 December 1965

3-196

TM-2624/100/

Date \_\_\_\_\_ programmer \_\_\_\_\_

Page 1 of 3

Job Number \_\_\_\_\_ For \_\_\_\_\_

1. Sequence Number of this Function

2. Card Code

Sub Code

3. Defined Input Area Designator (A-R)

(a) Index Register Assigned (10-14 or NN)

(b) Number of Slabs per Record (002-300)

(c) Number of Records per Block (01-99)

4. Defined Output Area Designator (A-R)

(a) Index Register Assigned (10-14 or NN)

(b) Number of Slabs per Record (002-300)

(c) Number of Records per Block (01-99)

5. Shall the Input Record from the Defined Input Area be moved to the Defined Output Area? (Y or N)

(If Y, it will be moved prior to any Operation by this Function)

(a) If Y, Shall any Excess Area in the Output Record be Zeroed prior to any Operation by this Function? (Y or N)

6. Shall the Overflow Flag be Cleared prior to Executing these Operations? (Y or N)

7. Number of Operations to be Performed (0-9, T, E)

(T = 10, E = 11) They are Performed in Order as Entered.

8. Operations:

Op. No.	Field A**				Field B**				Field C**			
	Area Desig.	Starting Location within Data Record	Lengths 1-8 Slabs	Operation* to be Performed	Area Desig.	Starting Location within Data Record	Lengths 1-8 Slabs		Area Desig.	Starting Location within Data Record	Lengths 1-8 Slabs	
1	B	046	1	E		A.F.7	4			1.2.0		
	27	28 29 30	31	32	33	34 35 36	37		38	39 40 41	42	
2	B	022	2	E	D	000	2			1.2.0		
	43	44 45 46	47	48	49	50 51 52	53		54	55 56 57	58	
3	B	049	1	I		A.1.7	4			2.3.0		
	59	60 61 62	63	64	65	66 67 68	69		70	71 72 73	74	

NOTE: The Sequence Number of the Next Operation to be Performed (Next Parameter Sheet, Cols. 72-74) MUST be entered regardless of the Number of Compute Function Operations Performed.

© 1965 THE NATIONAL CADASTRAL COMPANY  
U.S. GOVERNMENT PRINTING OFFICE

Figure 8. Compute Parameter Sheet, Page 1

For the Compute function, the Parameter Sheets come in sets of three. The second Parameter Sheet in the set is shown in Figure 9. You would fill out as many of these functions as needed for the particular job you want to do.

Figure 10 presents the third of the three sheets for the Compute function. Note that the documents are written in English. The programmer fills in numbers or answers questions to indicate the operating options to be utilized.

The next illustration (Figure 11) shows the updating transactions. In this particular example, there is a line for every different change. This is because, in production systems, most people desire and insist upon audit trails. Here, then, is a detailed audit trail showing everything that has happened to the Master File. This was the actual change listing as the new file was created.

The columns at the left, showing employee number and transaction code, indicate the different types of transactions given to this particular function. The next column lists the date of last activity, which provides a complete audit trail. It is not necessary to program for this audit trail data. The programmer just fills in a "yes" or "no" to indicate whether he wants the date of last activity.

Figure 12 shows the Control Report resulting from the original creation of the Master File. Note that this is not exactly the Control Report requested in the original outline of the problem. This report contains considerable detail on unit codes, persons employed, salaries, and number of organizational units. If the programmer does not want all of this detail, he can simply put an accumulate function into the program which will summarize each column. Such a report would contain the line across the bottom with summary totals only.

Some users say that at times they want the complete report and at other times they want just the summary line. To achieve this selectivity, it is possible to use Console Option Switches which can be set to deliver either type of report, according to the specific requirements of the moment.

Figure 13 shows what the actual report looked like when the Master File was updated. This is the listing for the updating of the three transactions, as requested.

Following this, of course, comes the new Control Report (Figure 14).

Now let us take a look at the Demand Report (Figure 15). It might be a little bit out of order. But, since the reports are all independent, it wouldn't make any difference. The Demand Reports are the ones which we are demanding from the Master File covering people whose records have certain characteristics. First, we have the program select the data sort, and then print the two Demand Reports.

Date \_\_\_\_\_ Programmer \_\_\_\_\_ Page 2 of 3

Job Number \_\_\_\_\_ For \_\_\_\_\_

## PROGRAM I.B.

D.E.M. 4.9.1

1.5.0

4.7

2

0

1. Sequence Number of this Function

2. Card Code

Sub Code

3. Operations (Continued)

Op. No.	Field A <sup>1</sup>			Operator <sup>2</sup> to be Performed	Field B <sup>2</sup>			Operator <sup>2</sup> to be Performed	Field C <sup>2</sup>		
	Area Design.	Starting Location within Data Record	Length 1-8 Slots		Area Design.	Starting Location within Data Record	Length 1-8 Slots		Area Design.	Starting Location within Data Record	Length 1-8 Slots
4	<input type="checkbox"/> F	0.4.9 9 10 11	1 12	<input type="checkbox"/> G	<input type="checkbox"/> A	3.4 15 16 17	2 18	<input type="checkbox"/> F	<input type="checkbox"/> A	2.3.0 20 21 22	3 23
5	<input type="checkbox"/> F	0.3.1 25 26 27	1 28	<input type="checkbox"/> F	<input type="checkbox"/> A	7.7 31 32 33	2 34	<input type="checkbox"/> F	<input type="checkbox"/> A	2.3.0 35 36 37	3 38
6	<input type="checkbox"/> F	0.2.6 41 42 43	2 44	<input type="checkbox"/> F	<input type="checkbox"/> D	0.0.8 47 48 49	2 50	<input type="checkbox"/> F	<input type="checkbox"/> A	1.7.0 51 52 53	3 54
7	<input type="checkbox"/> F	0.2.8 57 58 59	2 60	<input type="checkbox"/> F	<input type="checkbox"/> D	0.0.8 63 64 65	2 66	<input type="checkbox"/> F	<input type="checkbox"/> A	1.7.0 67 68 69	3 70

4. Sequence Number of the Next Function to be Performed.

2.3.0

- A = ADD; A + B → C
- S = SUBTRACT A - B → C
- M = MULTIPLY A × B → C
- P = DIVIDE A ÷ B → C
- G = Compare and Test G;  
If A > B, Go to C
- E = Compare and Test E;  
If A = B, Go to C
- L = Compare and Test L;  
If A < B, Go to C
- O = Test Overflow;  
If Overflow Flag is On, Go to C
- P = Place; Place the contents of Field A into Field C (Field B is ignored)
- R = Compare and if A ≤ B, go to C
- F = Set the flag in the location indicated by Field A (make an alphabetic space)
- C = Clear the flag in the location indicated by Field A (make an alphabetic zero)
- T = Test the flag in the location indicated by Field A and, if set, transfer to the location identified in Field C (Flag itself is not changed)
- J = JUMP

All Operations are as Defined in MCR 315 Programming Handbook except "F" or "R", which Operates as Defined Above.

(For Operations G, E, L, T, O, and J, Field C is Designated to be the Sequence Number of Another Function. In the Event of an Affirmative Answer to the Operation, Control is Transferred to that function. If, however, Field C is entered as "XX" (01 to 11) is One of the Eleven Operation Numbers of this Function, Control is Transferred to Perform that Operation and Proceed as Normal.)

- 1 If a Constant is Required by the Operation, a Number Sign (#) Entered as the Length will cause the Related Field to be used as the Constant.
  - Numeric if three numeric characters are entered.
  - Alpha if an "A" and two characters are entered.
- 2 If the Operation to be Performed is F, C, or T, then an R in the length position signifies the Right Flag in this Location, and an L signifies the Left Flag.
- 3 Work Areas may be used for Temporary Storage by entering a "W" in the First Position of the Field Location. (Range: W01-W99)
- 4 The Length must be Entered as Normal, However the Length of Each Work Area Available is 8 Slots. Each Area is Initially Zero.
- 5 "ACC" may be entered as the Field for Field A or Field C. For Field A this Entry assumes the Accumulator contains the Data to be operated upon by this Operation. For Field C this Entry assumes the Results of this Operation are to be Held in the Accumulator and not Stored. Lengths must be Specified as Usual.
- 6 (Field B may not Contain ACC).

Figure 9. Compute Parameter Sheet, Page 2

Page 3 of 3

Date \_\_\_\_\_ Program \_\_\_\_\_

Job Number \_\_\_\_\_ For \_\_\_\_\_

PROGRAM I.D.

D.E.M.4.0.1

1.5.0

4.7

3

0.0

1. Sequence Number of This Function

2. Card Code

Sub Code

3. Operations: (Continued)

Op. No.	Field A**				Field B**				Field C**			
	Area Design.	Starting Location within Data Record	Length 1-8	Operator to be Performed	Area Design.	Starting Location within Data Record	Length 1-8	Operator to be Performed	Area Design.	Starting Location within Data Record	Length 1-8	Operator to be Performed
8	B 1	0.3.0 10 11 12	2	E 1	D 1	0.0.8 23 24 25	2	E 1	3	1.7.0 26 27 28	2	3
9	B 1	0.3.2 26 27 28	2	E 2	D 1	0.0.8 29 30 31	2	E 2	3	1.7.0 32 33 34	2	3
10	B 1	0.3.4 42 43 44	2	E 3	D 1	0.0.8 45 46 47	2	E 3	3	1.7.0 48 49 50	2	3
11	B 1	0.3.6 56 57 58	2	E 3	D 1	0.0.8 59 60 61	2	E 3	3	1.7.0 62 63 64	2	3

Figure 10. Compute Parameter Sheet, Page 3

1 December 1965

3-200

TM-2624/100/00

DATE 09/13/65

EMP. NO.	T/C	DATE LAST ACTIVITY	DESCRIPTION	OLD	NEW
010261	11	/ /	NEW M/F RECORD		GUTTMAN, G. J.
010261	22	13/09/65	CHANGE ASSIGNED UNIT		2110
010261	23	13/09/65	CHANGE JOB CODE		1110
010261	24	13/09/65	CHANGE LEVEL		HEAD
010261	25	13/09/65	CHG. PRI. SKILL CODE		1110
010261	26	13/09/65	CHG. SEC. SKILL CODE		1135
010261	28	13/09/65	CHG. SEC. SKILL CODE		1130
010261	28	13/09/65	CHANGE SALARY	000000	017500
010261	29	13/09/65	CHANGE DATE OF BIRTH		111020
010261	31	13/09/65	CHANGE SEX CODE		M
010295	11	13/09/65	NEW M/F RECORD		GILCOTT, D. N.
010295	22	13/09/65	CHANGE ASSIGNED UNIT		2132
010295	23	13/09/65	CHANGE JOB CODE		1130
010295	24	13/09/65	CHANGE LEVEL		EMPL
010295	25	13/09/65	CHG. PRI. SKILL CODE		1130
010295	28	13/09/65	CHANGE SALARY	000000	009500
010295	29	13/09/65	CHANGE DATE OF BIRTH		022333
010295	31	13/09/65	CHANGE SEX CODE		M
011058	11	13/09/65	NEW M/F RECORD		JOHNSON, C. P.
011058	22	13/09/65	CHANGE ASSIGNED UNIT		2342
011058	23	13/09/65	CHANGE JOB CODE		3540
011058	24	13/09/65	CHANGE LEVEL		EMPL
011058	25	13/09/65	CHG. PRI. SKILL CODE		3540
011058	28	13/09/65	CHANGE SALARY	000000	008500
011058	29	13/09/65	CHANGE DATE OF BIRTH		071637
011058	31	13/09/65	CHANGE SEX CODE		M
011775	11	13/09/65	NEW M/F RECORD		OMLINGER, D. D.
011775	22	13/09/65	CHANGE ASSIGNED UNIT		2314
011775	23	13/09/65	CHANGE JOB CODE		3510
011775	24	13/09/65	CHANGE LEVEL		EMPL
011775	25	13/09/65	CHG. PRI. SKILL CODE		3510
011775	26	13/09/65	CHG. SEC. SKILL CODE		3745
011775	28	13/09/65	CHANGE SALARY	000000	008500
011775	29	13/09/65	CHANGE DATE OF BIRTH		030136
011775	31	13/09/65	CHANGE SEX CODE		F
012245	11	13/09/65	NEW M/F RECORD		KENT, M. J.
012245	22	13/09/65	CHANGE ASSIGNED UNIT		2133
012245	23	13/09/65	CHANGE JOB CODE		1150
012245	24	13/09/65	CHANGE LEVEL		EMPL
012245	25	13/09/65	CHG. PRI. SKILL CODE		1150
012245	26	13/09/65	CHG. SEC. SKILL CODE		3340
012245	28	13/09/65	CHANGE SALARY	000000	009800
012245	29	13/09/65	CHANGE DATE OF BIRTH		012516
012245	31	13/09/65	CHANGE SEX CODE		M
013060	11	13/09/65	NEW M/F RECORD		BROWN, G. E.
013060	22	13/09/65	CHANGE ASSIGNED UNIT		2340
013060	23	13/09/65	CHANGE JOB CODE		3220
013060	24	13/09/65	CHANGE LEVEL		EMPL
013060	25	13/09/65	CHG. PRI. SKILL CODE		3220
013060	28	13/09/65	CHANGE SALARY	000000	004500
013060	29	13/09/65	CHANGE DATE OF BIRTH		100346
013060	31	13/09/65	CHANGE SEX CODE		F
013090	11	13/09/65	NEW M/F RECORD		ELLARD, J. M.

Figure 11. Updating Transactions

1 December 1965

3-201

TM-2624/100/00

DATE 09/13/68

UNIT CODE	PERSONS EMPL.	SALARY	NO. ORG. UNITS
2000	2	34,000	1
2100	2	29,400	1
2110	2	21,700	1
2111	5	52,300	1
2113	4	48,000	1
2115	5	52,500	1
2120	2	18,500	1
2122	3	30,700	1
2123	5	44,500	1
2130	2	21,000	1
2131	4	42,500	1
2132	4	41,800	1
2133	5	48,300	1
2135	4	37,000	1
2190	1	22,000	1
2300	2	27,000	1
2310	2	19,000	1
2311	6	49,200	1
2313	6	49,850	1
2314	6	44,000	1
2320	2	14,500	1
2321	5	30,000	1
2322	4	23,000	1
2340	2	18,500	1
2341	3	24,600	1
2342	5	32,200	1
2350	2	17,200	1
2351	4	34,100	1
2353	5	44,700	1
TOTALS	104	969,050	29

Figure 12. Master File Control Report



1 December 1965

3-202

TM-2624/100/00

DATE 09/14/65					
EMP.NO.	T/C	DATE LAST ACTIVITY	DESCRIPTION	OLD	NEW
033144	81	13/09/65	REMOVED FROM M/P		
088657	26	13/09/65	CHG. SEC. SKILL CODE		2870
091152	28	13/09/65	CHANGE SALARY	008500	008500
				**END OF RPT**	

Figure 13. Master File Updating Report

1 December 1965

3-203

TM-2624/100/00

DATE 09/14/65

UNIT CODE	PERSONS EMPL.	SALARY	NO. ORG. UNITS
2000	2	34,000	1
2100	2	29,400	1
2110	2	21,700	1
2111	5	52,600	1
2113	4	48,000	1
2115	5	52,500	1
2120	2	18,500	1
2122	3	30,700	1
2123	4	39,000	1
2130	2	21,000	1
2131	4	42,500	1
2132	4	41,800	1
2133	5	48,300	1
2135	4	37,000	1
2190	1	22,000	1
2300	2	27,000	1
2310	2	19,000	1
2311	6	49,200	1
2313	6	49,850	1
2314	6	44,000	1
2320	2	14,500	1
2321	5	30,000	1
2322	4	23,000	1
2340	2	15,500	1
2341	3	24,600	1
2342	5	32,200	1
2350	2	17,200	1
2351	4	34,100	1
2353	5	44,700	1
<b>TOTALS</b>	<b>103</b>	<b>963,850</b>	<b>29</b>

Figure 14. Control Report

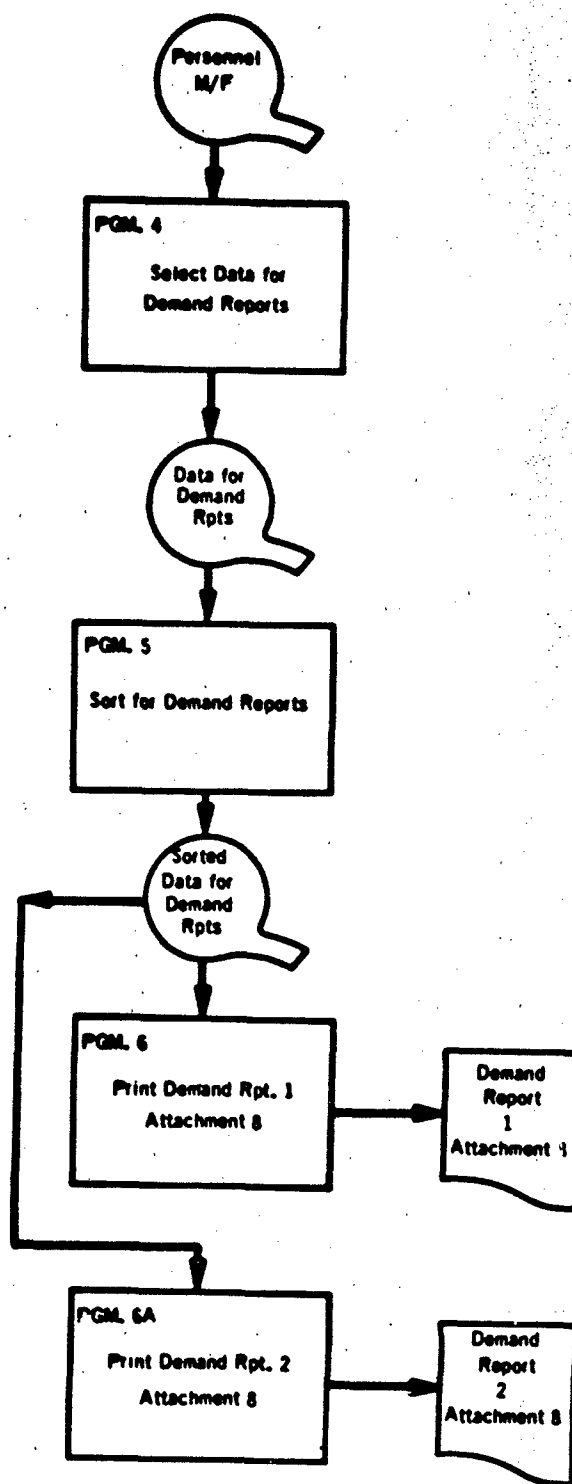


Figure 15. Demand Reports

Now, let's take a look at these programs in detail. Figure 16 tells us that we're going to get the personnel file. Now, one of the particular requests for this job was to determine whether the system could pull out the requirements for more than one Demand Report on a single pass of the file.

In this case, the answer is yes. That's what the zeroing does--up at the top. It clears out an area so that the users can put a flag on each record, or as many flags as are needed for as many reports as are to be produced.

Suppose you were asked for 18 reports. If you pull out each record for one person on each report, he may be on 16 out of your 18 reports. And you don't want to have to sort through 16 different copies of the same record. You only want to sort one copy by setting flags for the separate reports. We can tell a subsequent program which person may be applicable to any of the 16 of the 18 reports, instead of just one out of two, as in this case.

The two larger squares represent the functions which ask the Master File whether this particular record or person is desired for the particular report. And if you had three reports, you would have three of these squares. If you had four reports, there would be four, and so on. However, even if the person shows up on five reports, the program would show him only once in the sort.

In other words, this program selects the data for the Demand Report. Then a generated sort is run. The next program (Figure 17) determines which report is printed at any given time. If a system happens to have four printers, it is possible to generate four reports simultaneously as far as BEST is concerned. The programmer simply completes four REPORT function Parameter Sheets.

The actual Demand Report # 1, as it was printed on the high-speed printer, is shown in Figure 18. Note that there are multiple lines on the report for individual employees. It is worth noting also that the REPORT function automatically groups all items for a single employee on one page. If a total list of an individual's skills will not fit on a page, the program automatically carries all entries for this person over to the next page.

Figure 19 represents the complete Demand Report # 2.

After the Demand Report, we have the Periodic Report (Figure 20). This report requires information from both files. Its creation is the first time the files get together. At the top of the flow chart is the program which reads in the information from both files. It then selects the information from both files as necessary and puts it out on magnetic tape. The next program, represented by the second rectangle down from the top, sorts the data. The third rectangle depicts the program which prints the Section Report. The program for the Group Report is represented by the bottom rectangle.

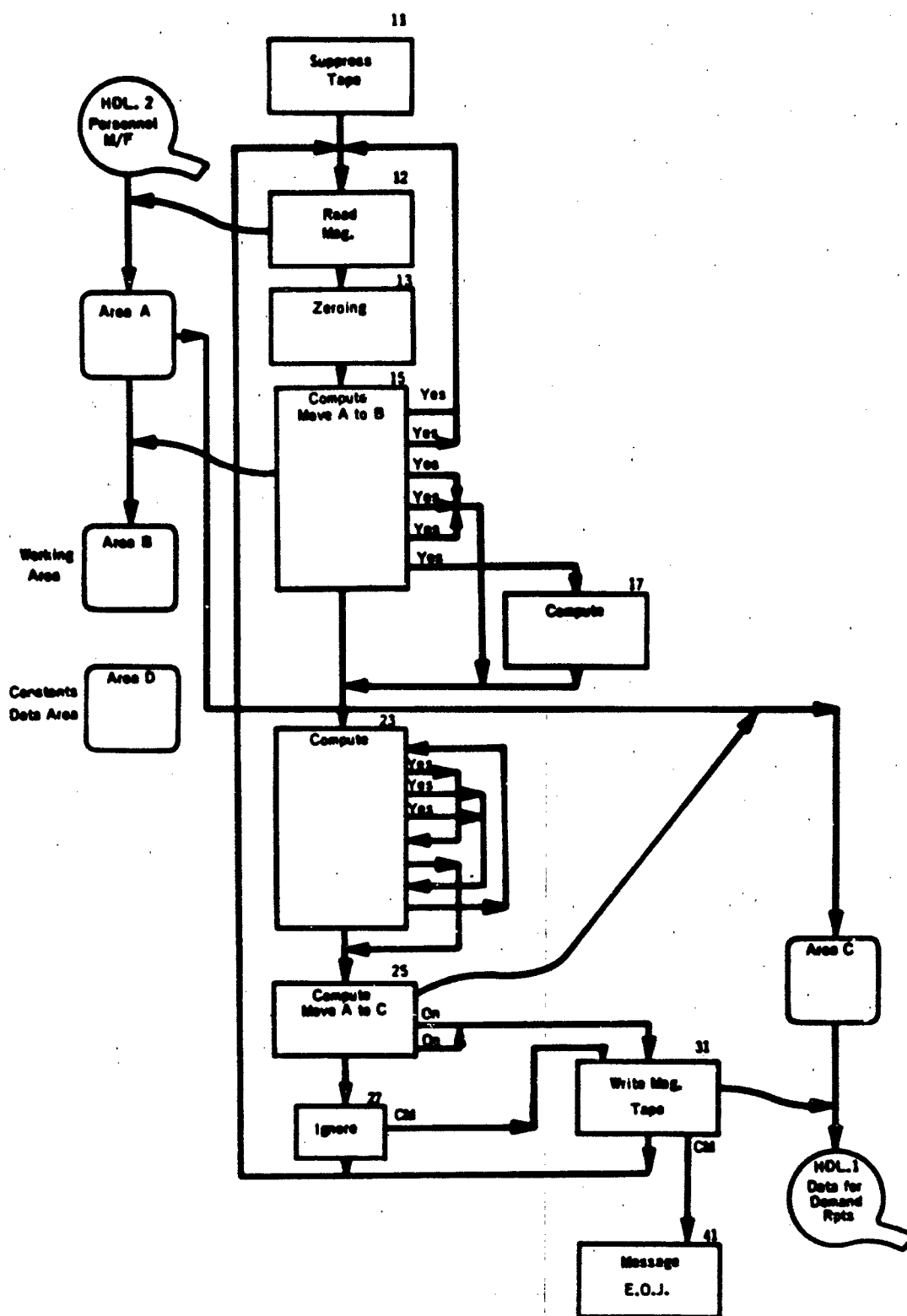


Figure 16. Select Data for Demand Reports

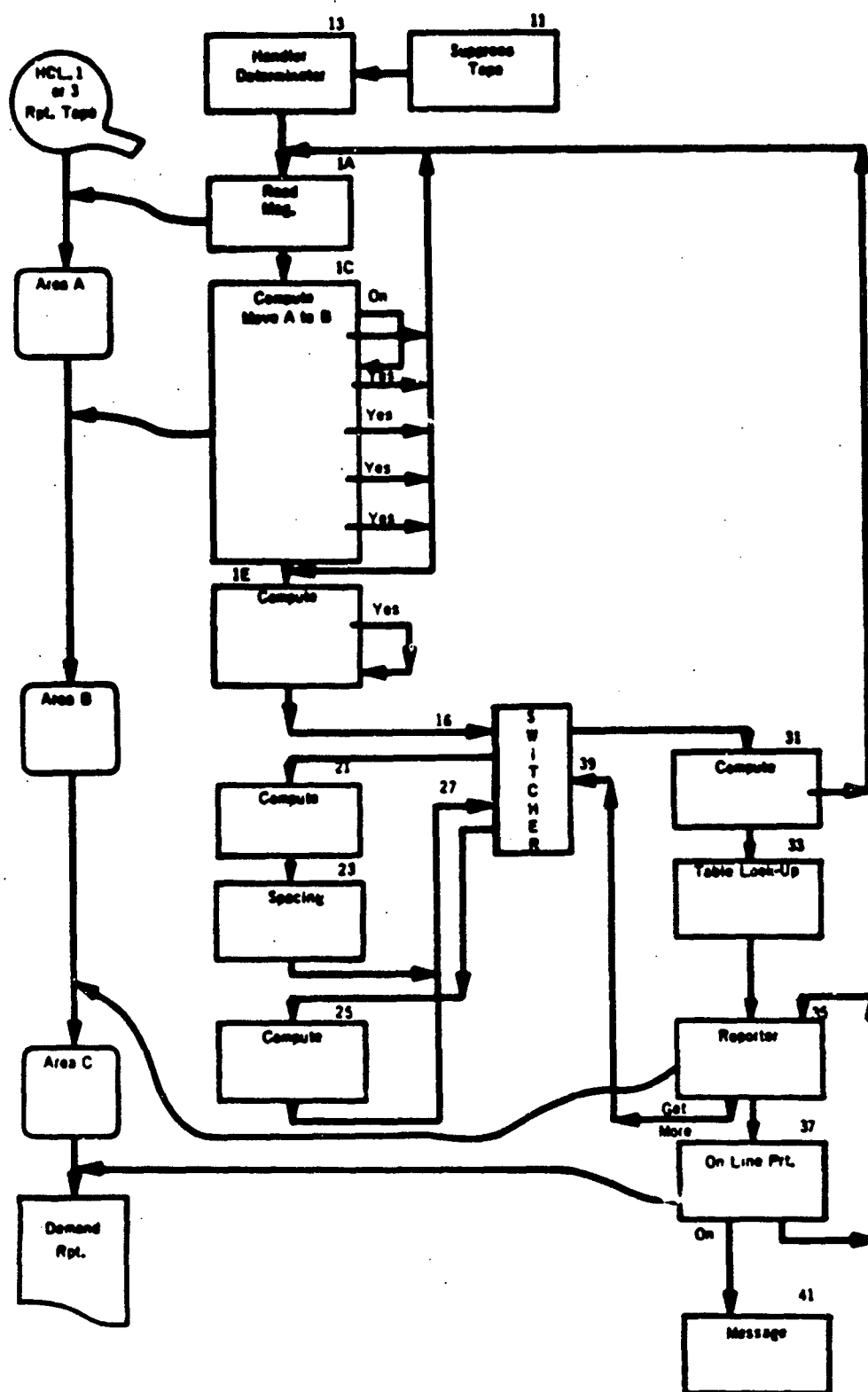


Figure 17. Demand Report Printing

1 December 1965

3-208

TM-2624/100/01

DATE 09/14/68	NAME	NUMBER	UNIT	SKILLS (PRIMARY SKILL LISTED FIRST)
	BOYD, M. V.	15052	2138	1351 MECH TECHN 3340 MACHINE OPR 7320 MAINT TECH (E) 3370 PAINTING OPR
	COATES, C. L.	81130	2313	3340 MACHINE OPR 3345 MILLING MACH OPR 3360 HEAT TREAT OPR
	FLETCHER, M. W.	21475	2133	1365 TOOL DESIGNER 3125 TOOL MAKER 3340 MACHINE OPR 7320 MAINT TECH (E)
	CORTON, R. A.	17590	2313	3340 MACHINE OPR 3345 MILLING MACH OPR 7320 MAINT TECH (E)
	GREEN, S. D.	34840	2313	3340 MACHINE OPR 3360 HEAT TREAT OPR 7320 MAINT TECH (E)
	LARNER, L. F.	78885	2353	7120 MAINT ENGR (E) 3340 MACHINE OPR 3125 TOOL MAKER
	LEE, R. E.	85657	2311	3340 MACHINE OPR 3570 DISPATCH ASST 3355 PLATING OPR
	LEVITT, P. S.	32924	2311	3340 MACHINE OPR 3345 MILLING MACH OPR 3360 HEAT TREAT OPR
	LITTLE, E. F.	42055	2311	3340 MACHINE OPR 3360 HEAT TREAT OPR 3550 EXPEDITER

Figure 18. Demand Report # 1

1 December 1965

3-209

TM-2624/100/00

DATE 09/14/68

NAME	NUMBER	UNIT	SKILLS (PRIMARY SKILL LISTED FIRST)
APGAR, A. J.	82802	2115	1130 ELEC ENGR
ARNETTE, L. J.	37113	2115	1130 ELEC ENGR
ETTINGER, G. J.	13581	2113	1130 ELEC ENGR
FRANCIS, G. C.	21777	2113	1110 SYSTEM ENGR 1130 ELEC ENGR
HENDERSON, R. G.	12130	2111	1130 ELEC ENGR
HASSARD, L. R.	36472	2132	1130 ELEC ENGR
RAYMOND, M. P.	28654	2123	1130 ELEC ENGR
SILCOTT, D. M.	10295	2132	1130 ELEC ENGR
SKINNER, J. P.	89876	2132	1130 ELEC ENGR
STADERMAN, D. K.	80823	2122	1130 ELEC ENGR

••END OF RPT••

Figure 19. Demand Report # 2



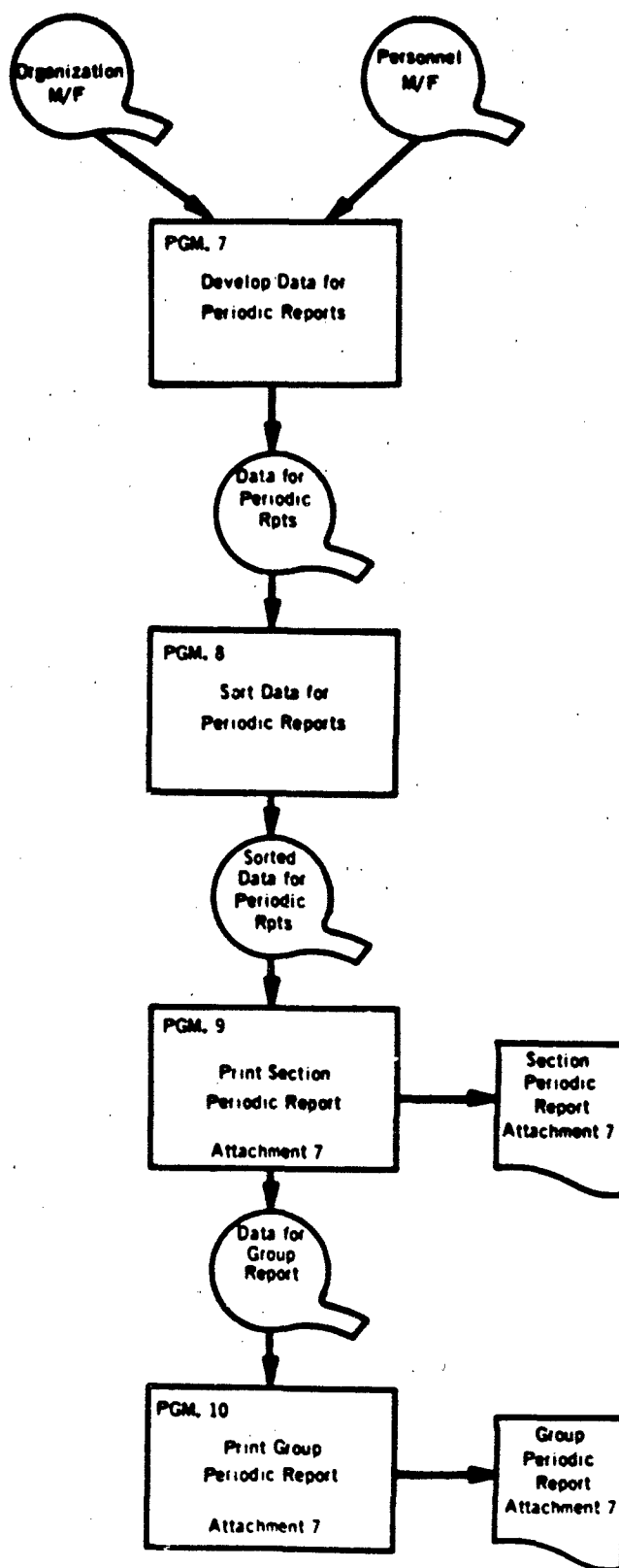


Figure 20. Periodic Report Generation

Figure 21 shows the flow chart both at the logic level and at the implementing level. In this particular case, there are 11 functions. To generate this entire selection program, including all 11 functions, you would have to fill out 14 sheets of paper. Each sheet would represent one 80-column punched card--or a frame on paper tape. The flow chart (Figure 21) represents about eight minutes of generator or compiling time and shows how the data is pulled off.

On the left side of the illustration is the Organization Master File. On the right side is the Personnel Master File. If you also had a Geographical Location Master File, you could have a third file on the right and you could manipulate and pull information from as many different files as necessary to complete the report.

Figure 22 shows a flow chart for the Periodic Report preparation. It is the only flow chart you have to write, or even look at or refer to, in conjunction with a program involving approximately 2,500 instructions. It is the only flow chart you have to assimilate if you are looking at somebody else's work.

This flow chart has about eight different types of functions. These functions, after you have been educated to know them, explain exactly what the memory load will do. Also, if you want to change this particular memory load at any level or at any place, you can. Any one of the arrows between any box is interruptable by any function--or by own coding if you so desire.

There is no such thing as a predetermined exit point in BEST. You are free to put exit points, decision points, or process points any place between any of the rectangular boxes (functions). The square boxes represent the data area which you probably would not fuss with too much.

This Periodic Report program does a number of things:

We pulled out the information from the personnel file and from the organization file and sorted it together. Since this report is going to be at the Section Level, if we summarize the two types of records--representing both budgets and actual people--together, we will come up with one record which covers both areas. By inserting a COMPUTE function into the program, we get the budget difference. In addition, the REPORT function generates column totals which will be carried forward as detail lines on the Group Report.

The printout of the actual Periodic Report (Figure 23) shows the actual deviations from the budget at the right.

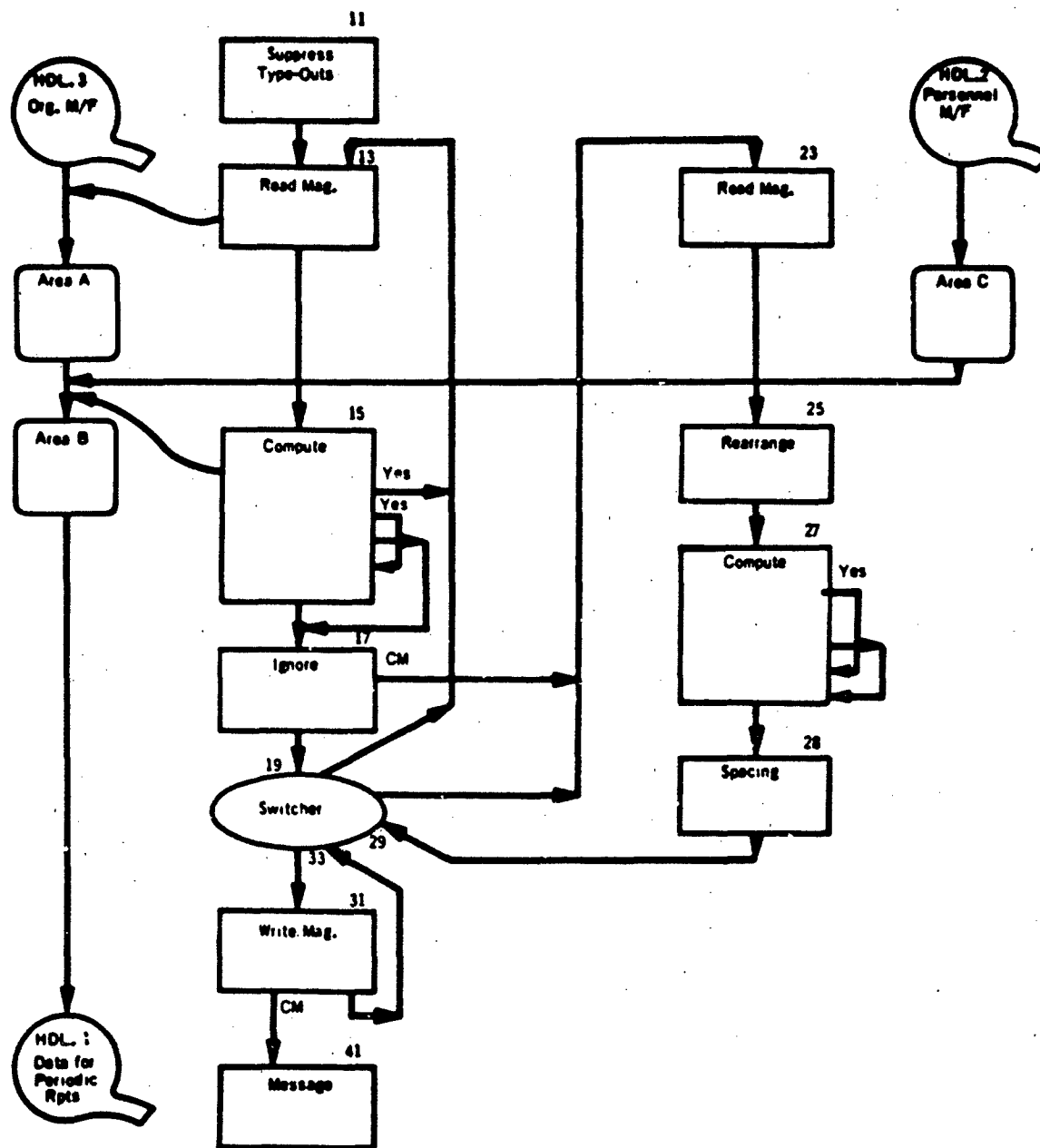


Figure 21. Development of Data for Periodic Report



DATE 09/14/65      ATTACHMENT 7 - PERIODIC REPORT 1 (SECTIONS)      PAGE 3

ORG. UNIT	2115	REPORTS TO 2110	ORG. NAME	PROD. SPEC. SECTION	JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	ACTUAL COMPLEMENT	ACTUAL SALARIES	DEVIATION FROM BUDGET	CR UNDER BUDGET
1110	2115		CHIEF					1	12,000	1	12,000		
1120	2115		MECH ENGR					1	11,000	1	11,000		
1130	2115		ELEC ENGR					2	11,000	2	21,500	10,500	
1330	2115		DRAFTSMAN					1	8,000	1	8,000		
TOTALS THIS UNIT								4	42,000	5	52,500	10,500	

Figure 23. Periodic Report Printout

Figure 24, then, is the Group Report which is generated by going back through the same program with only a few changes in the Parameter Sheets to pick up the total lines from the Section Report and incorporating them as detail lines in the Group Report.

As another by-product--and this is the only place where we interact between two of the requirements--we pulled out those particular sections which had under-budget or over-budget situations calling for further attention.

Figure 25 shows the major flow chart for the Exception Report. Again, it is the same data that was put into the Section Report in the first place because what we really want to do is pull out subsets of the same information. Over to the right is the data from the Group Report which tell us which sections to select to satisfy the criteria of being over or under budget for this month.

The flow chart in Figure 26 shows the program for the creation of the Exception Report. On the left is the sorted data. On the right is a file of only those sections that are within the requirements designated as overbudget or under budget for the month. Right in the middle is a function called COLLATE which operates on a match/merge basis to throw away the sections that we don't want.

Figure 27 is one of the pages from the Exception Report. The actual complement is bigger than authorized. This particular example is \$10,500 over budget.

The Hypothesis Report (Figure 28) is completely independent from the others, so it is best that we treat it that way. The Hypothesis Report really deals only with the Organization File. It has nothing to do with anything else. When we realize this, and realize also that the proposed organization is nothing other than records from another (hypothetical) personnel file, all we have to do is go to the Organization Master File and pull out two types of records. One type deals with the original organization. The other involves the "hypothetical" Personnel File.

Then, as the next function (Figure 29) shows, the program reads in the Organization Master File and puts out one record for the Organization Master File and another which treats organization records as though they were personnel records.

The Hypothesis Report (Figure 30) really is independent of all the others. So, you don't have to run all the other reports because there may be many different hypotheses to bring to bear on the situation. This particular Memory Load creates all of the different levels of reporting that are required. It has three different ACCUMULATE functions, each accumulating at its own level and taking as its input the output from the accumulation of the prior level. This is exactly the requirement of the problem.

DATE 09/14/65 ATTACHMENT 7 - PERIODIC REPORT 2 (GROUPS)

ORG. UNIT 2110 REPORTS TO 2100 ORG. NAME SYST. ENGR. GROUP

JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	ACTUAL COMPLEMENT	ACTUAL SALARIES	DEVIATION FROM BUDGET	CR UNDER BUDGET
1110	2110	MANAGER	1	17,500	1	17,500		
5210	2110	SECRETARY	1	4,200	1	4,200		
2111	2110	PROPOSAL SECTION	1	52,600	1	52,600		
2113	2110	ADV. SYSTEMS SECTION	1	49,300	1	48,300	1,000CR	
2115	2110	PROD. SPEC. SECTION	1	42,000	1	52,500	10,500	
TOTALS THIS UNIT			5	165,300	5	174,800	9,500	

Figure 24. Group Report Printout

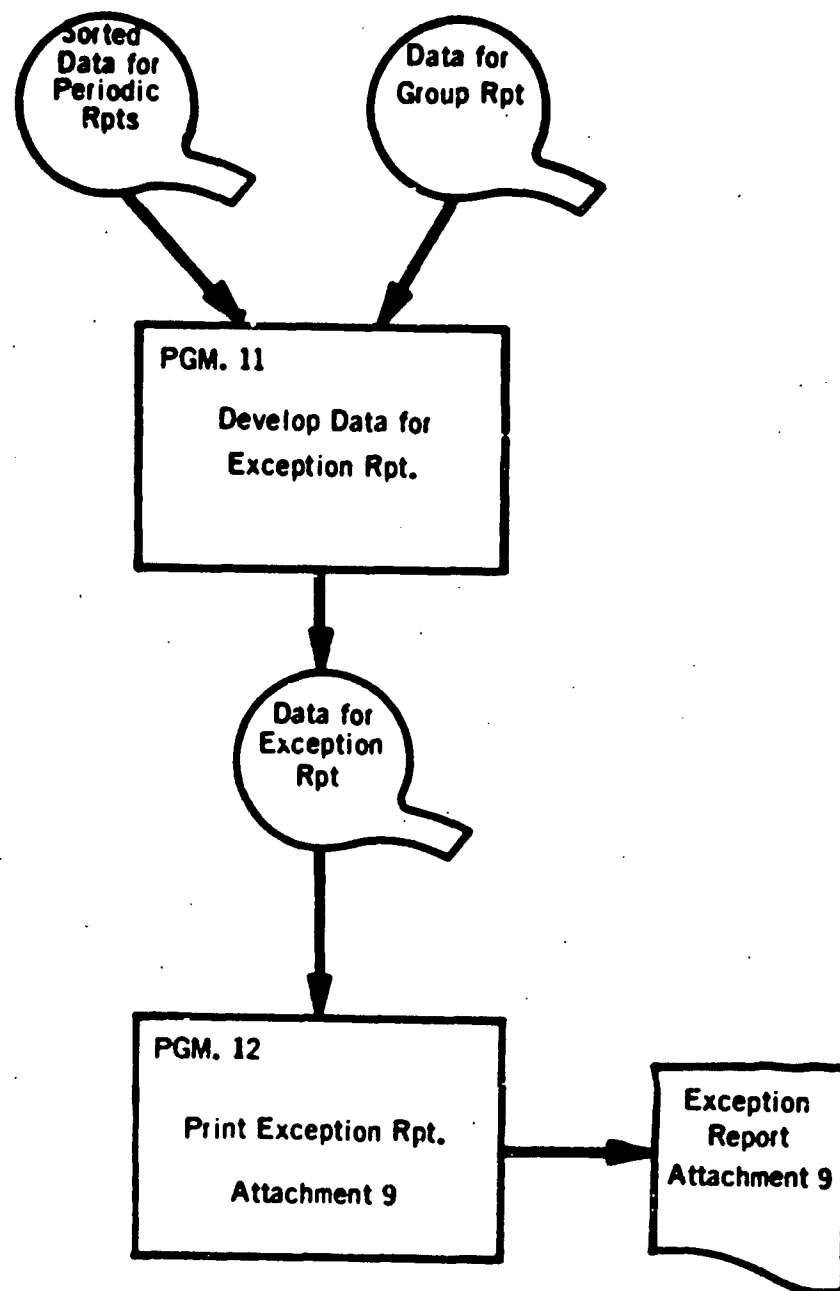


Figure 25. Exception Report



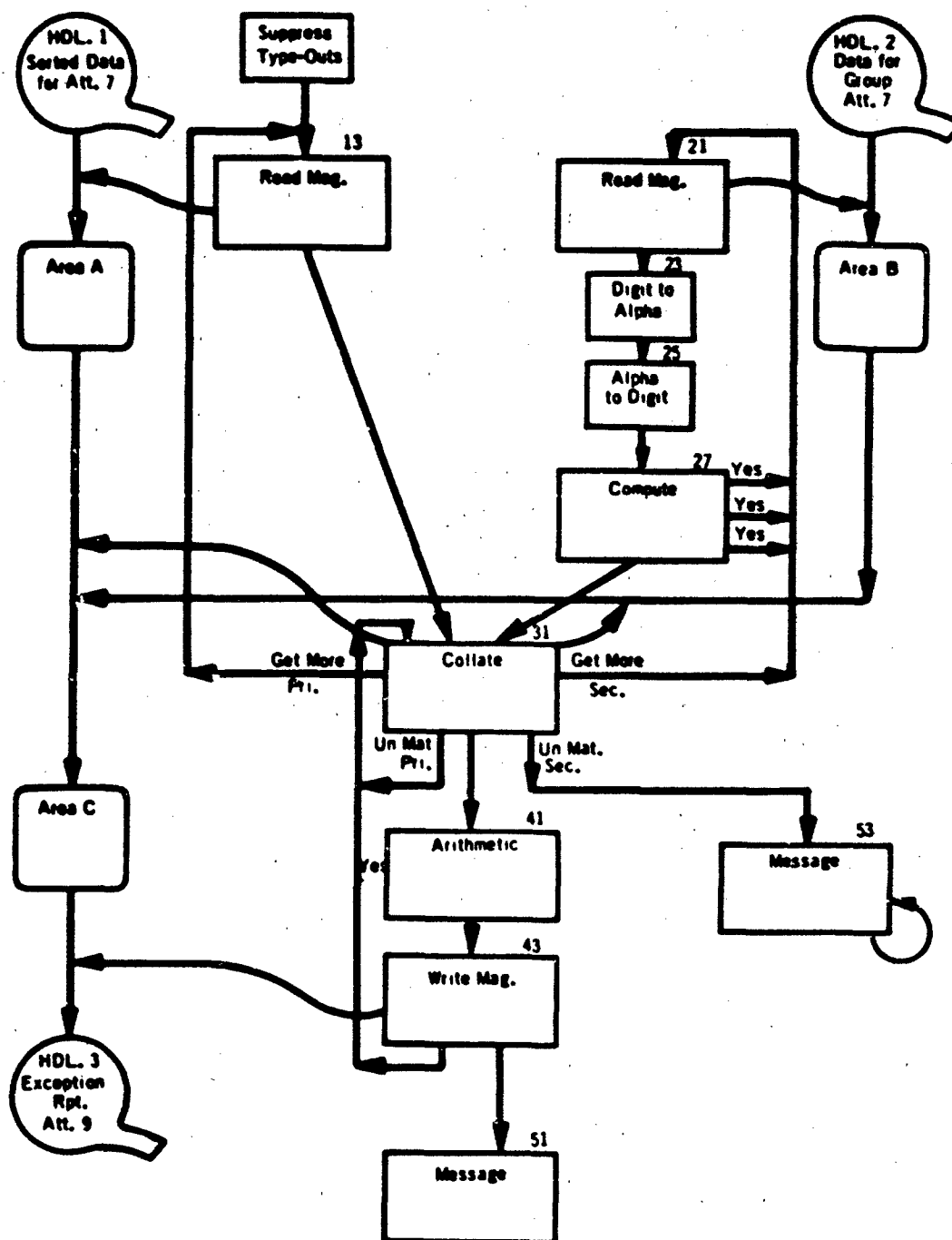


Figure 26. Selection of Data for Exception Report

DATE 09/01/65 ATTACHMENT 9 - EXCEPTION REPORT PRODUCED BY NCR'S BEST PAGE 2

ORG. UNIT 2115 REPORTS TO 2110 ORG. NAME PROD. SPEC. SECTION

JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	ACTUAL COMPLEMENT	ACTUAL SALARIES	DEVIATION FROM BUDGET	CR UNDER BUDGET
1110	2115	CHIEF	1	12,000	1	12,000		
1120	2115	MECH ENGR	1	11,000	1	11,000		
1130	2115	ELEC ENGR	1	11,000	2	21,000	10,000	
1320	2116	DRAFTSMAN	1	8,000	1	8,000		
TOTALS THIS UNIT			4	42,000	5	52,000	10,000	

Figure 27. Sample Page from Exception Report

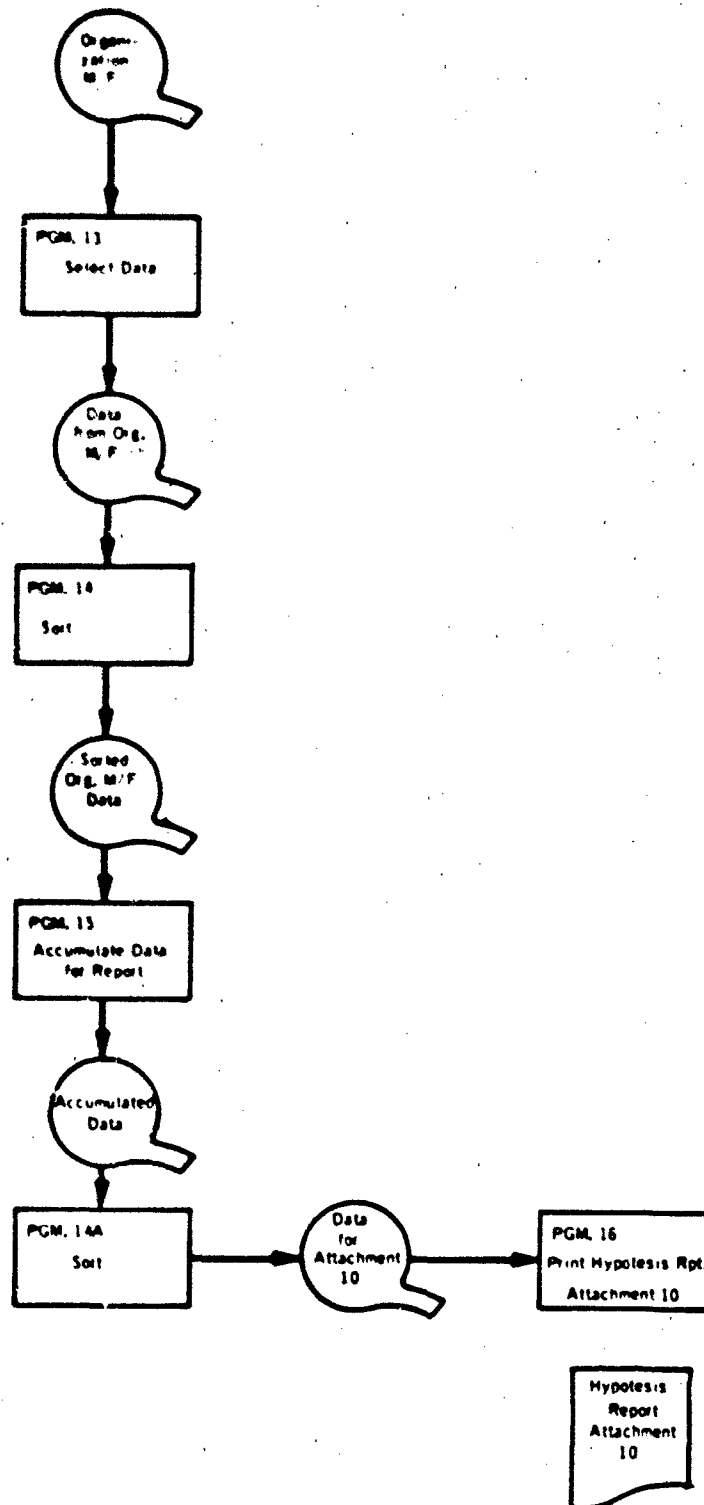


Figure 28. Hypothesis Report

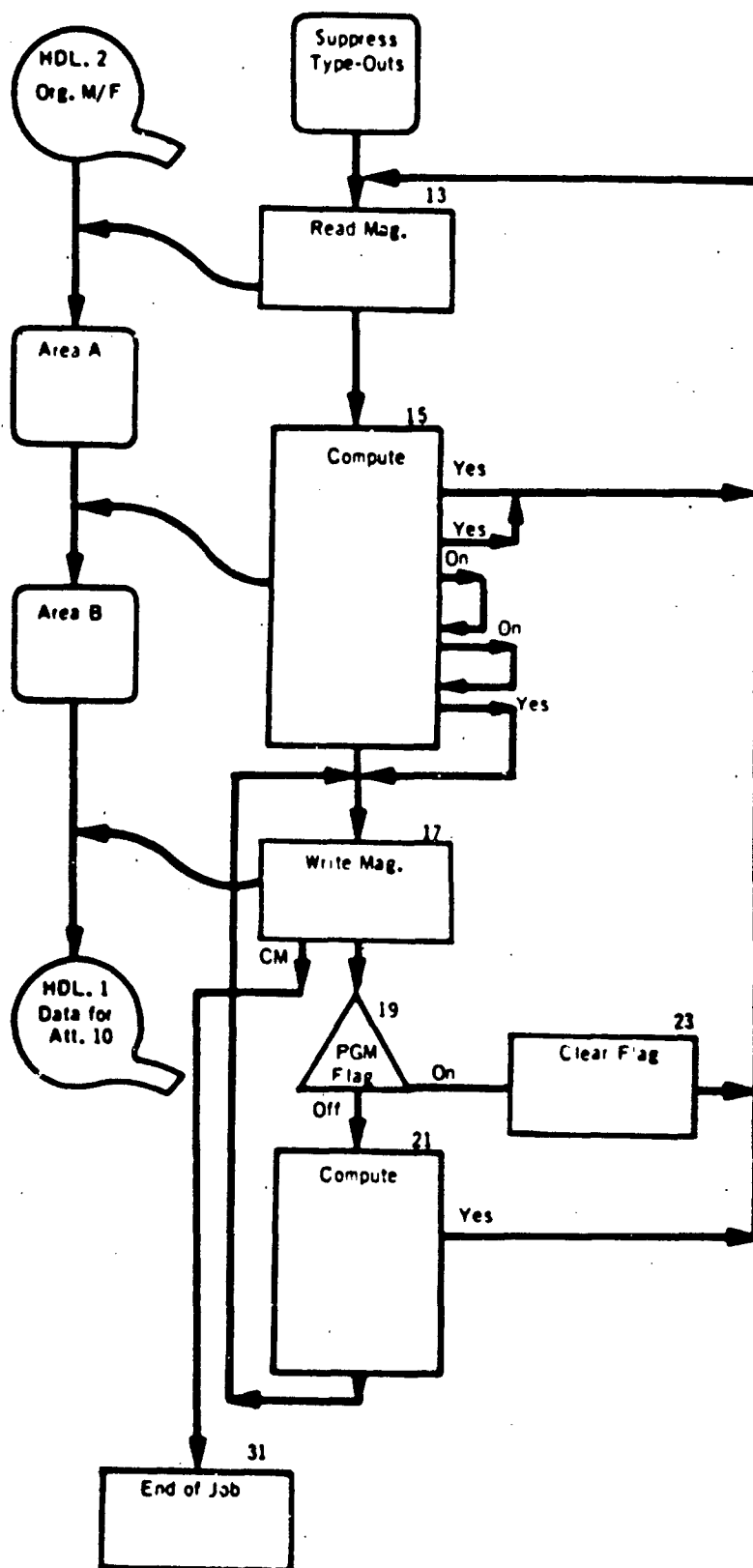


Figure 29. Selection of Data for Hypothesis Report

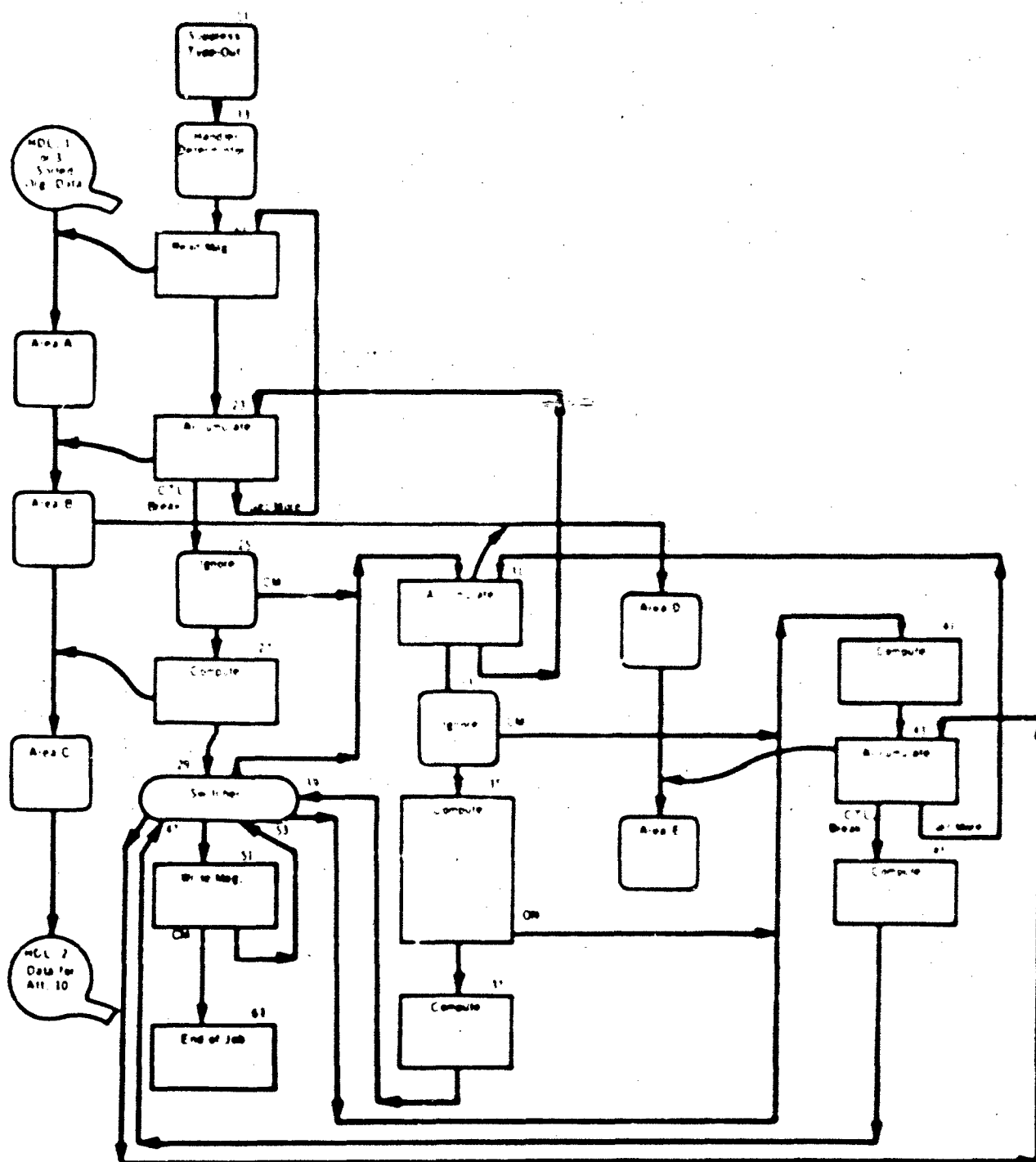


Figure 30. Development of Data for Hypothesis Report

This represents an excellent illustration of how you can stack these functions any way you want to in a Memory Load or program. You can do any arithmetic operation you want to. You can put these functions in any combination you want to. The only restrictions are your own ingenuity and the size of the memory.

Figure 31 shows the program that sets all the lines for all the reports and controls the printing of the Hypothesis Report. This is a typical reporting Memory Load, with eight separate data processing functions to do a complete printing job, including accumulating and a few other things.

Figure 32 shows the printing of the Hypothesis Report for Unit 2115. It shows, down at the bottom, that a draftsman left his particular section. The draftsmen, in effect, were transferred, because of the hypothesis, out of the section.

Figure 33 shows the section that the draftsmen were all transferred to. You will notice the drastic change in budget: \$87,000 vs. \$44,000.

The Group Report is shown in Figure 34. You will notice that there were 11 in this group and that there are now 15.

The last illustration (Figure 35) shows the total department and, thank heavens, it is still all there. We still have 48 people. We still have a \$510,000 budget. But the department has been realigned slightly.

Now, if you want to test another hypothesis, all you have to do is change the criterion of reporting to indicate which individuals are to report to what job codes and which job codes report to what groups. Then, just put it through the same basic program and you will realize the same type of results.

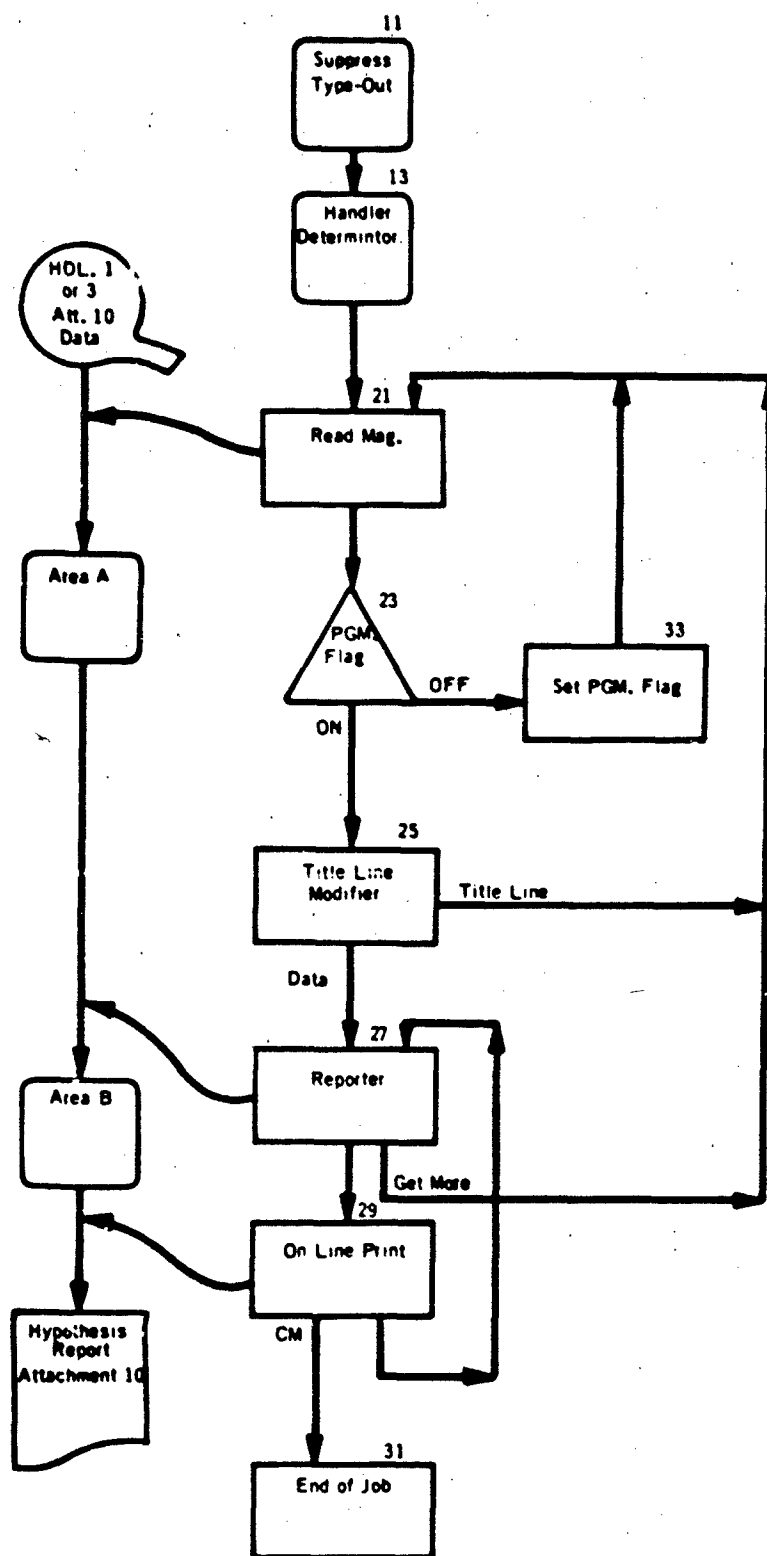


Figure 31. Control of Printing of Hypothesis Report

1 December 1965

3-225

TM-2624/100/00

DATE 09/01/65		ATTACHMENT 10 - HYPOTHESIS REPORT		PRODUCED BY ACR, S BEST	
ORG. UNIT 2115		REPORTS TO 2110		ORG. NAME PROD. SPEC. SECTION	
JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	PROPOSED COMPLEMENT
1110	2115	CHIEF	1	12,000	1
1120	2115	MECH ENGR	1	11,000	1
1130	2115	ELEC ENGR	1	11,000	1
1330	2115	CRAFTSMAN	1	8,000	1
TOTALS THIS UNIT			4	42,000	3
					34,000

Figure 32. Printing of Hypothesis Report



1 December 1965

3-226

TM-2624/100/00

DATE 09/01/65 ATTACHMENT 10 - HYPOTHESIS REPORT PRODUCED BY ACR,S BEST  
 ORG..UNIT 2123 REPORTS TO 2120 ORG. NAME COMPONENT STDS. SEC.

JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	PROPOSED COMPLEMENT	PROPOSED BUDGET
1120	2123	CHIEF	1	11,000	1	11,000
1120	2123	MECH ENGR	1	9,500	1	9,500
1130	2123	ELEC ENGR	1	10,000	1	10,000
1320	2123	DRAFTSMAN	1	8,500	6	51,000
5520	2123	FILE CLERK	1	5,500	1	5,500
TOTALS THIS UNIT			5	44,500	10	87,000

Figure 33. Hypothesis Report Example

1 December 1965

3-227

TM-2624/100/00

DATE 09/01/65		ATTACHMENT 10 - HYPOTHESIS REPORT		PRODUCED BY NCR, S BEST	
ORG. UNIT 2120		REPORTS TO 2100		ORG. NAME STDS. ENG. GROUP	
JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED BUDGET	PROPOSED COMPLEMENT
1120	2120	MANAGER	1	14,500	1
5210	2120	SECRETARY	1	4,000	1
2122	2120	SYSTEM STDS. SECTION	4	39,200	3
2123	2120	COMPONENT STDS. SEC.	5	44,500	10
TOTALS THIS UNIT			11	102,200	15
					136,200

Figure 34. Group Report Example

1 December 1965

3-228

TM-2624/100/00

DATE 09/01/65		ATTACHMENT 10 - HYPOTHESIS REPORT		PRODUCED BY NCR/S BEST		
ORG..UNIT 2100		REPORTS TO 2000 ORG. NAME DEVELOPMENT DEPT.				
JOB CODE	SECTION	NAME	AUTHORIZED COMPLEMENT	AUTHORIZED SUDGET	PROPOSED COMPLEMENT	PROPOSED BUDGET
1110	2100	MANAGER	1	24,000	1	24,000
2110	2100	SECRETARY	1	5,400	1	5,400
2110	2100	SYST. ENGR. GROUP	15	165,300	13	148,800
2120	2100	STDS. ENGR. GROUP	11	102,200	15	136,200
2130	2100	COMPONENT ENGR GROUP	19	192,000	17	174,500
2190	2100	CHIEF SCIENTIST	1	22,000	1	22,000
TOTALS THIS UNIT			48	510,900	48	510,900

Figure 35. Total Department Report

## QUESTION:

Do you compile and store programs, or do you always regenerate your programs?

## ANSWER:

The system itself generates one program at the beginning, and then we can store it and reuse it as often as we want to without regeneration. Incidentally this system does not occupy any core while it's operating. The entire machine is available to the user--whoever the user is. So we can store it and have all the programs and everything stacked on an entire library. All an operator has to do now to get this entire set of reports is push the "go" button, and it goes through the entire system, in about 25 minutes. That includes generating the entire master file at the beginning, doing all the sorts, going on to the next program, and waiting for the operator to push the "go" button every now and then.

## QUESTION:

Is the system generalizable to other makes of equipment or is it limited to the 315 in particular?

## ANSWER:

The system, as an idea, can be used on any piece of equipment--it just hasn't been yet. It's a basic idea.

## QUESTION:

What was the man-hour requirement to prepare the problem for input to the computer?

## ANSWER:

The man hours were for two different men. I spent about 10 hours on it. The person who did the detail work--flow charting, etc.--spent 80 hours. Now, realize one thing. The detail work is now finished. It does not have to be documented any further. The man can quit and go to the next job. The flowcharts that you saw obviously were prepared by technical draftsmen, but they were prepared from the flow charts that he made in the thinking stage. He filled out the forms you saw. Of course he filled out 100 or so in addition to the ones I showed. (Every time there's a different record layout, it has already been filled out and is already in document form and in a book.) The man who prepares flow charts can go on to the next job, and anybody in the future, including himself, can go back and pick the charts up, change them if desired, and add to them or delete parts of them, because the things are set up in a format. Another thing which is also happening--we have a program which is

about ready for release which will take the parameter receipts for the parameter cards and write flow charts, similar to the ones you saw up here, on a computer and on a printer. This means that once the parameter has been filled out, the flow chart is already in a uniformly communicable form. Therefore, all of the file-people processes and all the record layouts--in a sense, all of the logic--will be contained in one little deck of parameter cards. Each one of these programs undertakes from about 15 punched cards to perhaps 100 to 120 cards. So one deck represents the entire program. I can communicate the whole thing with just one deck.

QUESTION:

Can the system generate new files as required, or is the system limited to the files that were stated?

ANSWER:

This particular system, as parameterized, cannot. But if you desire to change to a different file layout or additional ones, you fill out another three parameter sheets and then can issue a whole different one.

## INTEGRATED DATA STORE

Submitted by: Charles W. Bachman  
Consultant, Product Planning  
General Electric of Phoenix, Arizona

## INTRODUCTION

The Integrated Data Store (IDS)<sup>1,2,4,5\*</sup> is a computer language designed to facilitate the organization, storage, maintenance, and retrieval of information using a mass memory storage medium. IDS is not an information storage and retrieval system of and by itself. Rather, it is a set of tools which enables its user to design and implement an information storage and retrieval system specifically suited to the user's requirements. The IDS language is used in conjunction with COBOL to provide logical record-processing capabilities which are not specifically available in the COBOL language. IDS/COBOL programs have access to the capabilities of both languages. The IDS language provides data description elements which declare the existence of master/detail relationships between records (Figure 1). These relationships are implemented through chain link (list processing) techniques which create circular or ring structure. Each record in the chain contains the address of the next.

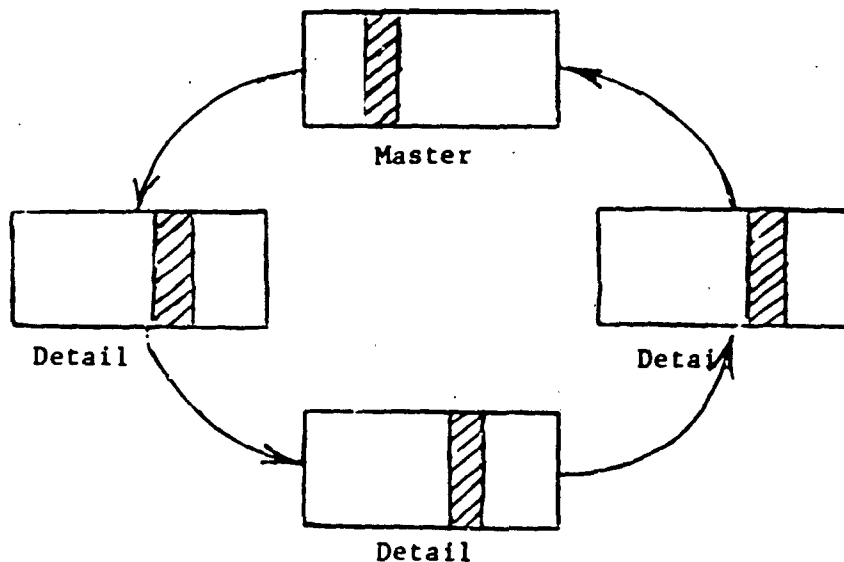


Figure 1. Schematic Diagram

\*Numbers in superscript refer to references presented on page 3-275.

The master record links to the first detail record. The first detail links to the second detail, etc. Finally, the last detail in the chain links back to the master record, closing the ring structure. Optionally, records may contain the address of the prior record and/or master record of the chain.

The procedural portion of the language contains the following verbs that actuate record-processing functions:

STORE  
RETRIEVE (in seven different flavors)  
MOVE TO WORKING-STORAGE  
MODIFY  
DELETE

These commands may be freely mixed with standard COBOL, and they automatically create, maintain, and delete the chain structure specified in the data description. The IDS system is designed for programs which operate in either a batch or direct access mode. The various retrieval commands permit the user the best use of both serial and random access to records.

#### BACKGROUND AND CURRENT STATUS

The Integrated Data Store is a product of the General Electric Company's corporate research into integrated business systems. It was developed by the same organization and component that developed decision or structure tables.<sup>6</sup> Its earliest roots extend back to the system of Report and File Maintenance Generators<sup>7</sup> developed by the General Electric Company at Hanford, Washington. That development work was culminated in the development of 9PAC (later 90PAC), the Report and File Maintenance Generators for the IBM 709. The 9PAC ideas, which were well suited to serial files, were further generalized to handle the capabilities and help solve problems associated with random access storage.

The Integrated Data Store is presently operational on the GE-200 series (215, 225, 235) computers. The GE-200 implementation was first used on a production basis in early 1964. It requires 8K of memory for a minimum use. It is currently being used at four computer sites with new installations in the planning stage. The GE-200 IDS differs from the GE-400 and 600 IDS in that it was integrated with the general assembly language for the 200 rather than with COBOL. This integration caused some clumsiness in the use of the language. However, the performance was not affected. The GE-400 and 600 IDS/COBOL is scheduled for availability and use late in 1965.

#### MAJOR FEATURES

The Integrated Data Store is a software system that provides the ability to store and retrieve records of any length within a mass memory. Each record type is defined by COBOL data descriptions to contain a specified number of fixed size fields and to participate in a specified number of chains. Different

record types have different fields, participate in potentially different chains, and therefore have a different length.

The IDS record-structuring capabilities permit any number of record types to be organized into any number of master/detail relationships. The meaning of this will become clear as the IDS solution to the case problem is studied.

The Integrated Data Store completely upsets some of the conventional views. "Input" and "output" seem to reverse. Storing and modifying records are input functions because they add to or update the information base. Retrieval and deletion are the output functions because they copy or remove information which is in the information base. Expressed in another way, the data base becomes the center of interest. The computer is an auxiliary that reads cards or their equivalent information "in" so as to assist the input process for the Integrated Data Store. The computer also assists in the output process of extracting information from the Integrated Data Store and forwarding it out of the system in terms of reports or answers to inquiries. To carry the analogy further, the computer is just the tool used by the user's program to facilitate his processing of the data base.

The computer and the data base can be studied from another point of view, the data storage capacity. The largest modern computer has the ability to hold a few million characters of data. Data bases of a billion characters or more are well within the capacity of their companion data storage devices (magnetic discs, strips, or cards). Therefore, the computer's view of a data base is much like a motorist's view of a city. He can see what is in front of and behind him, and a little bit to each side. It takes a lot of driving around a city to get much of an overview of what exists and where it is. The ability of a given (or written) program to study an "information city" also depends upon methodically searching through the data base until it finds what it is looking for. In traversing a city, we used to be limited by the traditional gridiron of streets where, at every corner, we had a chance to continue or turn right or left. The development of one-way streets and "no left turn" corners has made driving more complicated. Extra planning is necessary to reach your destination. The construction of modern expressways has helped to speed us to our destination. However, an additional element of planning is now necessary to determine which expressway exit to use to minimize travel time (seldom distance) to reach your objective. In the same way, driving instructions are necessary to learn to drive a computerized procedure efficiently over a data base. Did you ever get lost in a data base? It's not hard. You may need all the tricks that the Indians used to blaze a trail through an unknown wilderness. As data bases grow, they, like cities, will have traffic problems because more than one procedure will want to drive down the streets at the same time. Therefore, a comprehensive data base management system will have to set up stop lights and traffic cops to avoid collisions.

The Integrated Data Store uses chains to link together related records. These chains are the highways which the computerized procedures will use to seek



information. The design of these highways is largely based upon the subsequent information searches that are planned. The test case prepared for group study by the Second Symposium on Computer-Centered Data Base Systems provides an excellent area to study the use of information highways.

Figure 2 is a road map of the entire network of avenues (chains) designed to facilitate the storage, maintenance, and retrieval of information. This type of graphical presentation is called a data structure diagram. However, it is expressed in a condensed form which will need some explaining.

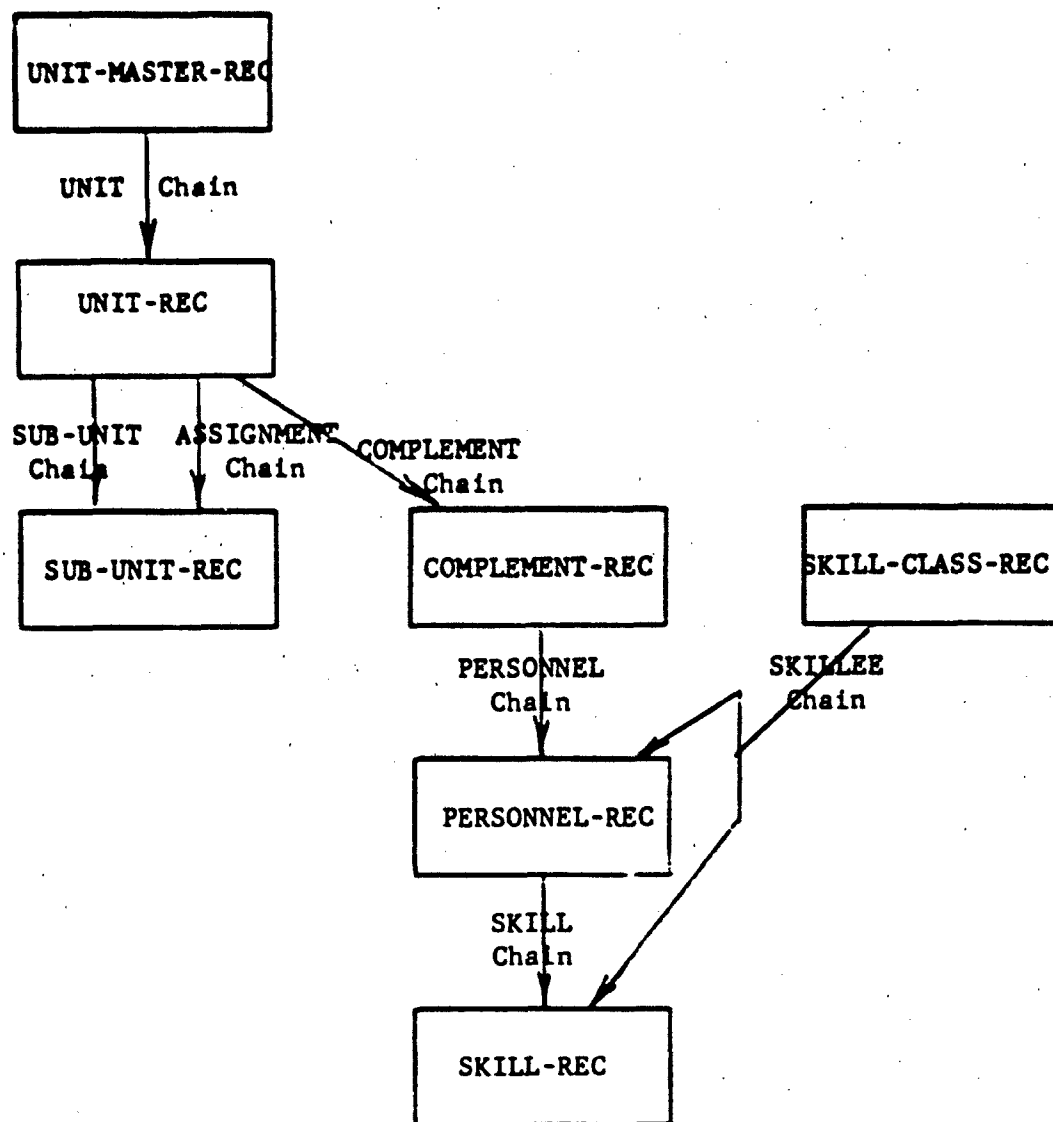


Figure 2. Data Structure Diagram



This chain has been specified as a sorted chain. The reader will note that UNIT-REC records are in ascending UNIT-CODE sequence. Sequencing is one of the chain-order parameters that can be specified.

Figure 4 illustrates the IDS/COBOL data description elements necessary to specify these records and chains. These data descriptions will be extended later as other records and chains relationships are declared.

```
01 UNIT-MASTER-REC.  
    98 UNIT CHAIN MASTER;  
        CHAIN-ORDER IS SORTED,  
  
01 UNIT-REC.  
    02 UNIT-CODE; SIZE 4 NUMERIC.  
    02 REPORTING-UNIT; SIZE 4 NUMERIC.  
    02 ORG-NAME; SIZE 20 ALPHANUMERIC.  
    02 TOTAL-BUDGET; SIZE 7 NUMERIC.  
    98 UNIT CHAIN DETAIL;  
        SELECT CURRENT MASTER;  
        ASCENDING KEY IS UNIT-CODE;  
        DUPLICATES NOT ALLOWED.
```

Figure 4. Partial Data Description Listing

Once this data description is established, the Integrated Data Store will insert each new UNIT-REC record in the UNIT chain according to its UNIT-CODE value. The IDS will also prevent the storage of a new UNIT-REC record if the record contains a UNIT-CODE which is equal to some already existing UNIT-CODE.

The data structure diagram in Figure 5 is a condensed way of graphically stating the key elements contained in the data description (Figure 4).

In Figure 5, blocks are used to signify the existence of record types, and arrows are used to illustrate the chain relationship between records. The arrow points from the master record to the detail record. There is no attempt in the diagram to say how many of each type of record exists.

Using Figure 3 as a specific information highway map, or Figure 5 (its condensed form), a programmer can now begin to think of going for a drive to see what he can see. The IDS procedural verb "RETRIEVE" is the steering wheel.

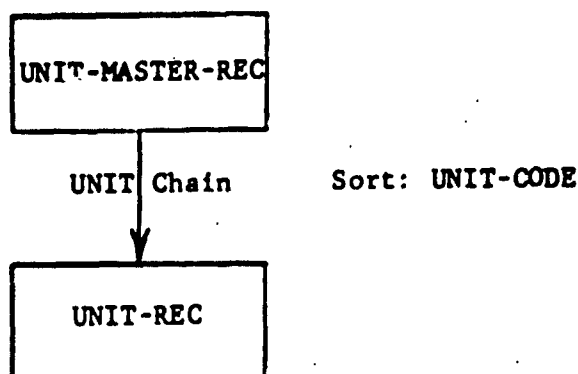


Figure 5. Data Structure Diagram

There are three forms of the RETRIEVE verb established for the purpose of accessing records that are linked together.

. RETRIEVE NEXT RECORD OF chain-name CHAIN:

This statement will access the next record of the named chain. Of course, the next record is relevant to the current record of the chain. Therefore, there must be a current record that was accessed by a previous chain retrieval command, or some other command. Underlined words are required, the others are for ease in human interpretation.

. RETRIEVE PRIOR RECORD OF chain-name CHAIN:

This statement will retrieve the prior record in the chain relative to the current record in the chain. The execution of the command will follow one of two courses: it will step immediately backward to the prior record, or it will run around the chain until it finds the prior record. The second course is obviously time-consuming and not a process to be used frequently. The long route around is only selected if the system does not know the address of the prior record. The system will know this prior record under two conditions: if the previous command which accessed this chain were a RETRIEVE NEXT, or if a "prior" chain field has been specified when the data description was established. The phrase "LINKED TO PRIOR" will cause each record in the chain to be initialized with an additional chain field containing the address of the prior record.

. RETRIEVE MASTER RECORD OF chain-name CHAIN:

This statement will retrieve the master record of the named chain. The execution of this statement will follow one of two courses: it will jump immediately to the master record, or it will run around the

chain until it finds the master record. The long route around is only selected if the system does not know the address of the master. The system will know this master address under two conditions: if the master record of this chain has been previously accessed, or if a "master" chain field has been specified when the data description was specified. The phrase "LINKED TO MASTER" will cause each detail record to be initialized with an additional chain field containing the address of the master record.

The flow chart (Figure 6) shows how the RETRIEVE NEXT command can be used to access each of the UNIT-REC records in the file for reporting purposes and exit when they have all been processed.

A report prepared by a procedure similar to the flow chart in Figure 6 will be in UNIT-CODE sequence. Specific tests can be established and coded in COBOL to determine whether the unit, whose UNIT-REC record was retrieved, is to be included in the report. For example, the Section, Group, and Exception Budget Reports could all be processed using this logic while using different selection tests to determine whether the retrieved UNIT-REC record is to be accepted for further processing (or reporting purposes).

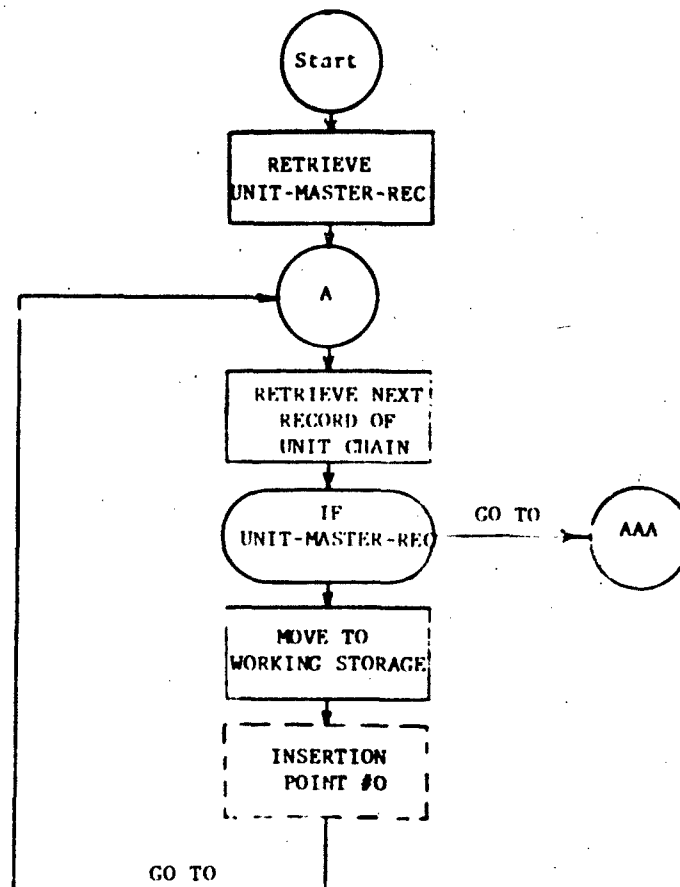


Figure 6. Flow Chart

All IDS/COBOL program listings shown represent partial programs in the sense of a complete COBOL program and are designed for illustrative purposes. They could not be compiled until all of the necessary parts of a COBOL program were assembled. Figure 7 is the listing of the IDS/COBOL program used to carry out this procedure.

```
      START.  
*      OPEN DATA-BASE.  
      OPEN OUTPUT REPORT-FILE,  
      INITIATE ALL.  
*      RETRIEVE UNIT-MASTER-REC.  
*      A.  
*      RETRIEVE NEXT RECORD OF UNIT CHAIN;  
*      IF UNIT-MASTER-REC GO TO AAA; ELSE  
*      MOVE TO WORKING-STORAGE,  
NOTE  REPORT FUNCTION INSERTION POINT NUMBER 0.  
      GO TO A.  
      AAA.  
      TERMINATE ALL.  
      CLOSE REPORT-FILE.  
*      CLOSE DATA-BASE.  
      STOP RUN.
```

\* IDS statements.

Figure 7. Partial Program Listing

The UNIT-REC records could be accessed in sequence using another route through the data base. This route is based upon a different set of chains. These chains were established for a different purpose, namely for the purpose of associating units in the hierarchy of unit organization. Because the assignment of unit codes was based upon the organization hierarchy, the hierarchy chains can be used to produce sequenced reports. In the same sense, the UNIT-CODE sequence chain (UNIT chain) can be used to produce organization hierarchy reports. The chain structure diagram in Figure 8 illustrates the records and chains used to represent the hierarchical information.

This structure of one record type with two different chains linking to another record type is the IDS way of defining hierarchical information structures. It is used to structure parts-list data, PERT diagram data, etc.

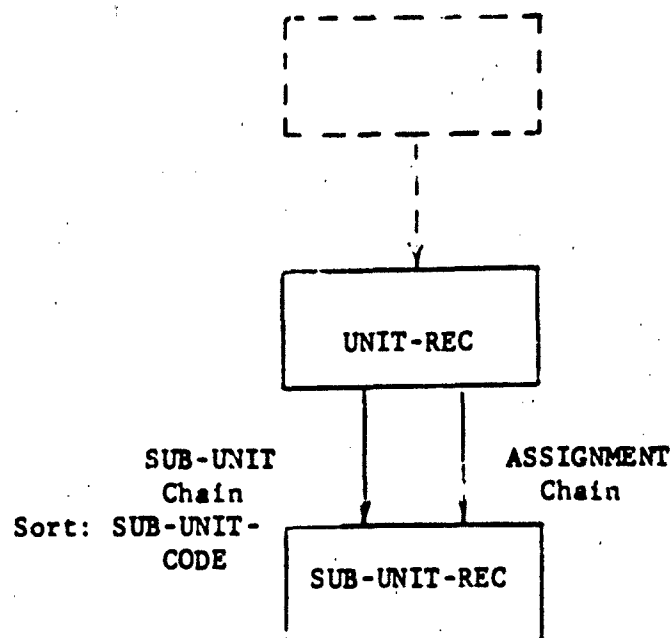


Figure 8. Data Structure Diagram

The actual data in the data base problem would create a specific structure that is illustrated in Figure 9. This figure shows a schematic diagram dealing with the division, UNIT-CODE 2000, and all of its lower level units.

The data description listing for Figures 8 and 9 may be seen as part of the complete data description presented on the first page of Figure 30.

In the diagram, the mechanism of the double chain linkage between UNIT-REC records is illustrated. Specifically, the UNIT-REC record 2000 for the division is connected downward to Units 2100 and 2300, the Development and Operations Departments. The UNIT-REC record 2100 for the Development Department is linked downward to the UNIT-REC records 2110, 2120, 2130, and 2190. UNIT-REC record 2110 is linked downward to UNIT-REC records 2111, 2113, and 2115. The flow chart in Figure 10 illustrates the procedure required to use the hierarchy chains, "SUB-UNIT" and "ASSIGNMENT" to produce a hierarchy sequenced report, which is also UNIT-CODE sequenced.

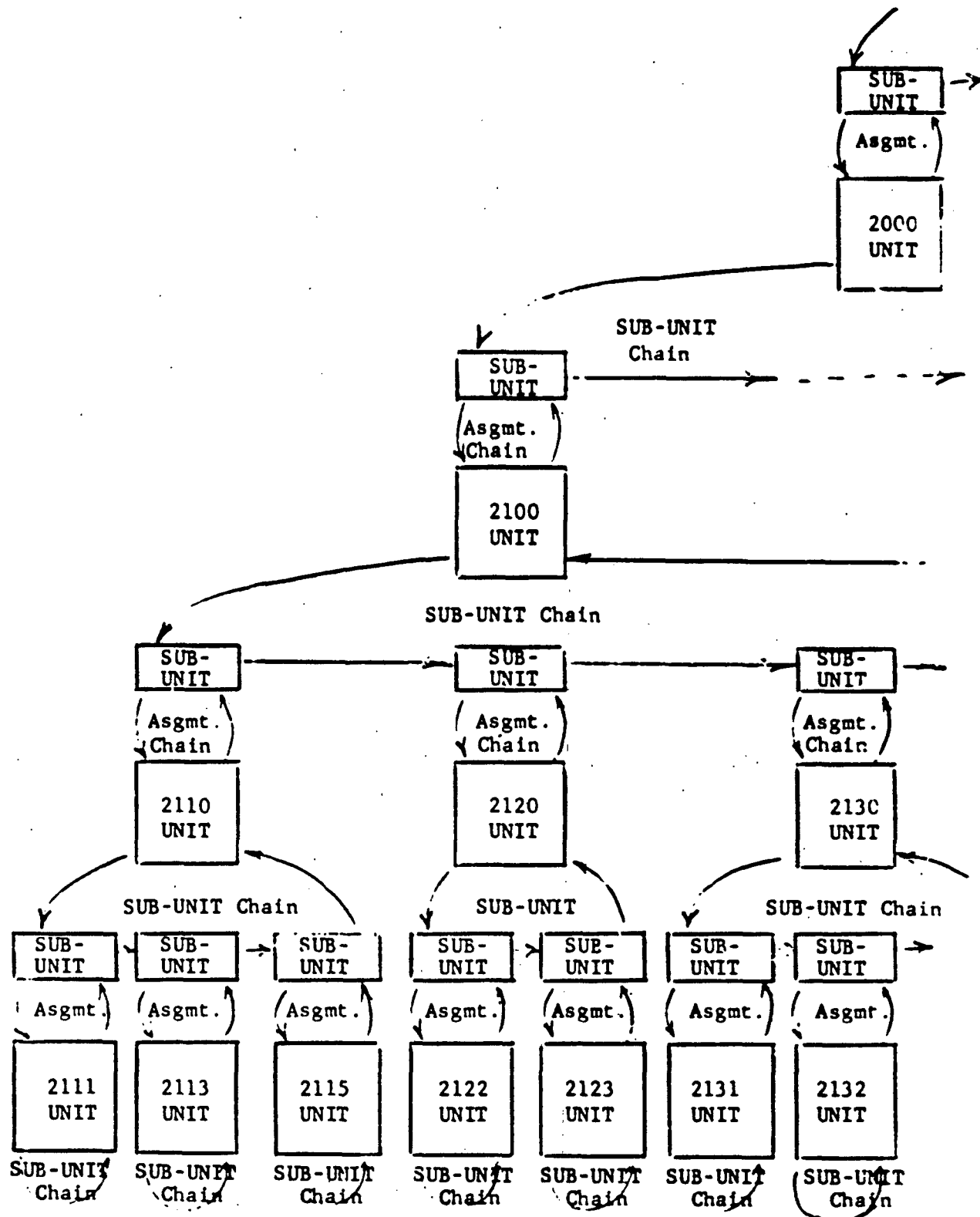


Figure 9. Schematic Diagram



If the reporting-function coding is inserted at Insertion Point 1, the report will come out in hierarchical order with each unit followed by its lower level units. The Group Budget Report is this type of report. If the report coding is inserted at Point 2, the report will then be produced with lower level units followed by their higher level units in the total position. Figure 11 is the IDS/COBOL coding equivalent to Figure 10.

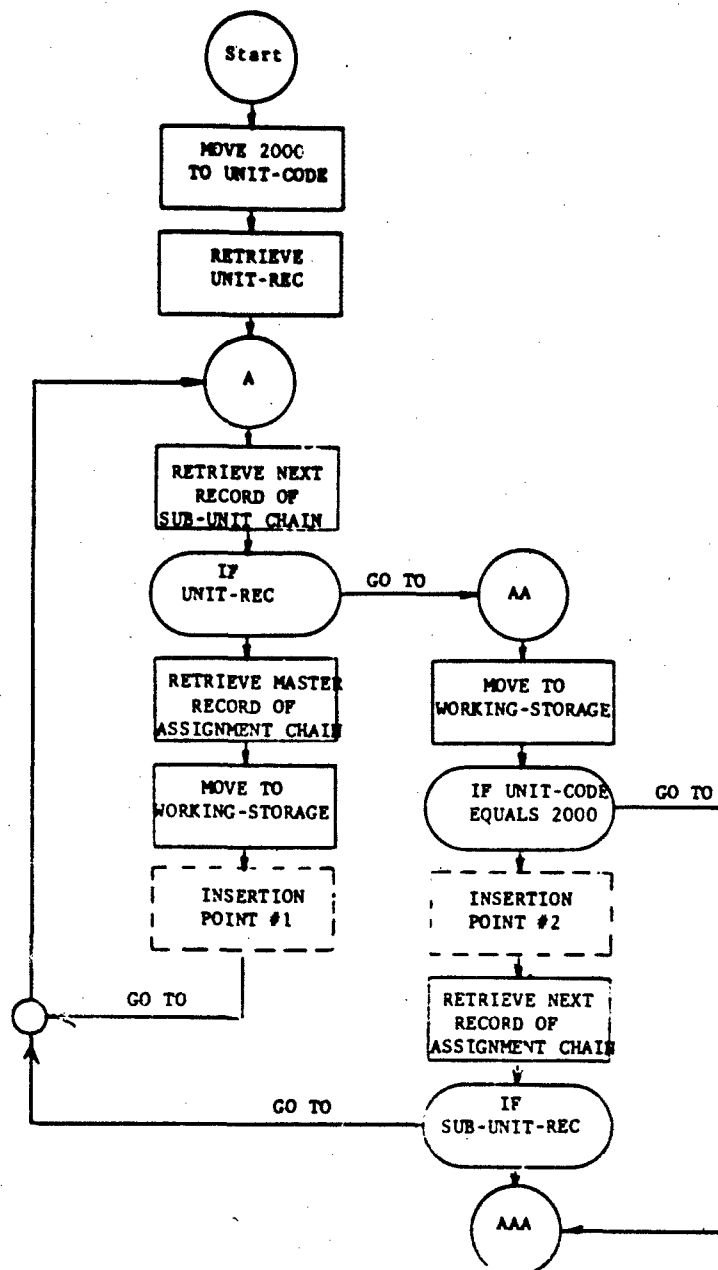


Figure 10. Flow Chart

If either the UNIT chain or the SUB-UNIT chain had been specified with a "prior" chain field, a sequenced record could have been produced in descending UNIT-CODE sequence by going around the chains in the "prior" direction. Substitution of "RETRIEVE PRIOR RECORD OF UNIT CHAIN," or "PRIOR OF SUB-UNIT CHAIN" into the coding in Figure 7 or 11, respectively, will change the direction of the retrieval of procedures.

```

      START.
*      OPEN DATA-BASE.
      OPEN OUTPUT REPORT-FILE.
      INITIATE ALL.
      MOVE 2000 TO UNIT-CODE,
*      RETRIEVE UNIT-REC.
      A.
*      RETRIEVE NEXT RECORD OF SUB-UNIT CHAIN
*      IF UNIT-REC MOVE TO WORKING-STORAGE; GO TO AA.
*      RETRIEVE MASTER RECORD OF ASSIGNMENT CHAIN;
*      MOVE TO WORKING-STORAGE,
      NOTE REPORT FUNCTION INSERTION POINT NUMBER 1.
      GO TO A.
      AA.
      IF UNIT-CODE EQUALS 2000 GO TO AAA.
      NOTE REPORT FUNCTION INSERTION POINT NUMBER 2.
*      RETRIEVE NEXT RECORD OF ASSIGNMENT CHAIN;
      IF SUB-UNIT-REC GO TO A,
      AAA.
      TERMINATE ALL.
*      CLOSE DATA-BASE.
      CLOSE REPORT-FILE.
      STOP RUN.

```

\* IDS statements

Figure 11. Partial Program Listing

#### SECTION BUDGET REPORT

The Section Budget Report calls for certain information about each unit, its authorized organization complement, and its actual complement.

The overall report is to be by unit in UNIT-CODE sequence. Either of the two methods (Figures 6 or 10) described for retrieving the UNIT-REC records in UNIT-CODE sequence could be used for this report. The actual processing of each UNIT-REC and its detailed information will be discussed from this point on.

Every unit has certain job classes and quantities of personnel authorized. Employees have been assigned to these job classes. The data structure designed to accept this type of information is shown in Figure 12. Two new types of

records have been established, the COMPLEMENT-REC and PERSONNEL-REC records, to handle the jobs and people. One PERSONNEL-REC record will be stored for each employee in the system. Each employee is assigned to some job in some unit. The COMPLEMENT-REC represents an authorized job category for a unit. Each PERSONNEL-REC is linked to a COMPLEMENT-REC through the PERSONNEL chain to represent the employee's assignment to that job. The COMPLEMENT-REC record is the master of the PERSONNEL chain. Every COMPLEMENT-REC record is linked in turn to the unit that it represents through the COMPLEMENT chain. The UNIT-REC record is the master of the COMPLEMENT chain. Using these three records and two chains, every employee is related to an authorized complement, and each authorized complement is related to a particular unit.

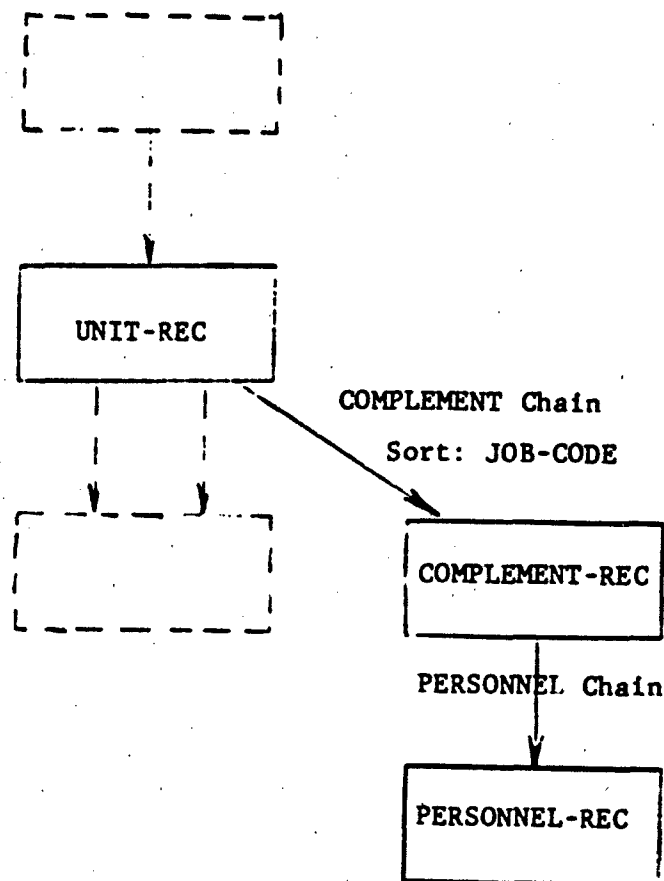


Figure 12. Data Structure Diagram

Figure 13 is a schematic diagram of the actual records which would be created to store the job and employee data concerning Unit 2115, the Product Specification Section.

The budget reports in Attachment 5, 7, 9 and 10 of the case study all call for a listing of the QUANTITY and TOTAL-SALARY data contained in the complement records for the "authorized complement," and then a summary of the actual personnel count and salaries for the "actual complement" portion of the reports.

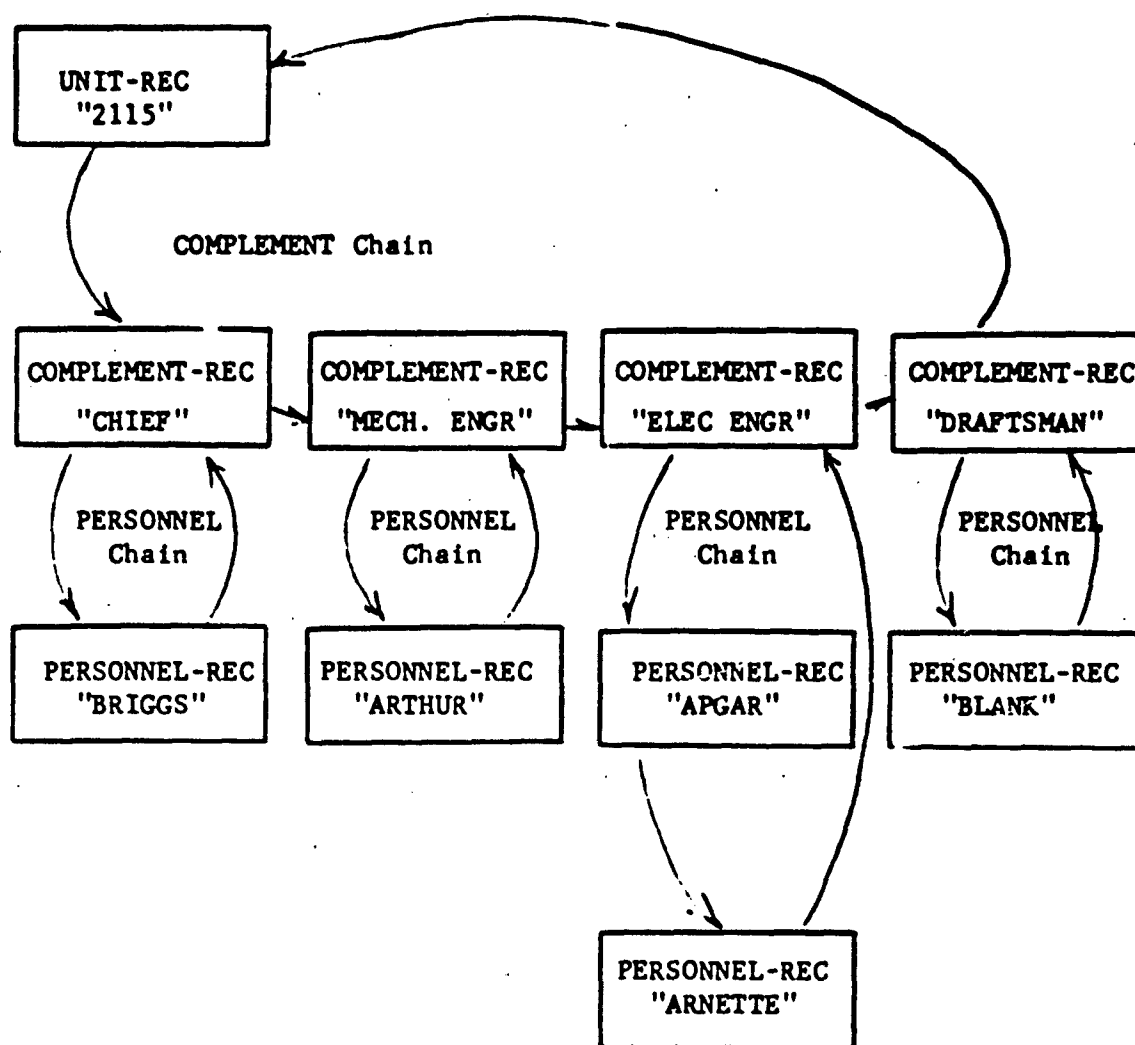


Figure 13. Schematic Diagram

Figure 14 is the flow chart of the report preparation procedure for the Section Budget Report. Figures 15 and 16 are subroutines (F thru K, and L thru N) used by the report preparation procedure.

Figure 14 contains the procedure for printing the report header (LINE-A), the intermediate report header (LINE-E), and the final total line (LINE-F). The figure also shows two calls on the subroutine "F thru K" (Figure 15).

The subroutine "F thru K" shown in Figure 15 retrieves each COMPLEMENT-REC record in the COMPLEMENT chain and prints the JOB-CODE, POSITION-TITLE, QUANTITY, TOTAL-SALARY (GENERATE LINE-B). This subroutine is so organized that it will call upon subroutine "L thru N" to derive the actual quantity and actual total salary from the PERSONNEL-REC records linked to the COMPLEMENT-REC records if requested. The value stored in the field named ACTUAL-COMPLEMENT controls whether the authorized or actual figures will be collected. When the entire COMPLEMENT chain has been traversed, the total print line (LINE-D) is printed.

Subroutine "L thru N," shown in Figure 16, is called whenever an actual budget figure is needed. It traverses the PERSONNEL chain adding up the salaries of the personnel and adding one to the count for each PERSONNEL-REC retrieved. When the entire PERSONNEL chain has been traversed, control is returned to subroutine "F thru K."

The reader should note that the repertoire of each organizational unit is independent of all others. Therefore, the system will produce information for one or more organizational units with equal facility and economy. In other words, this problem solution is good for both batch and on-demand reporting.

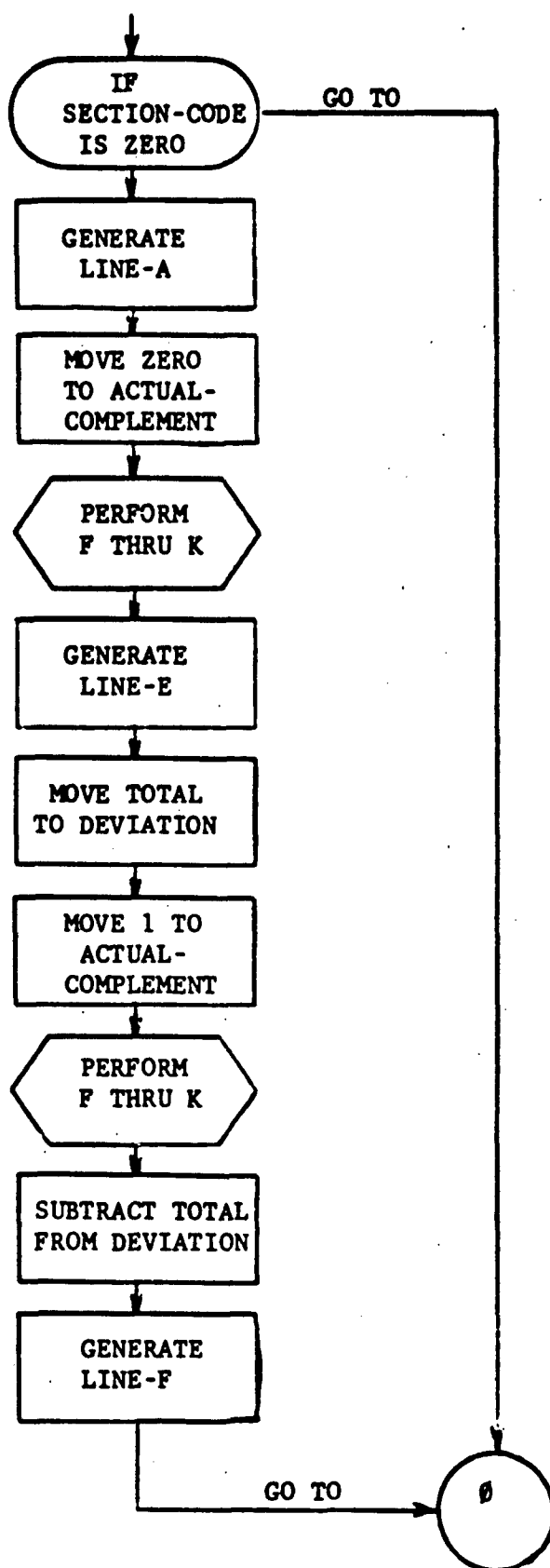


Figure 14. Flow Chart

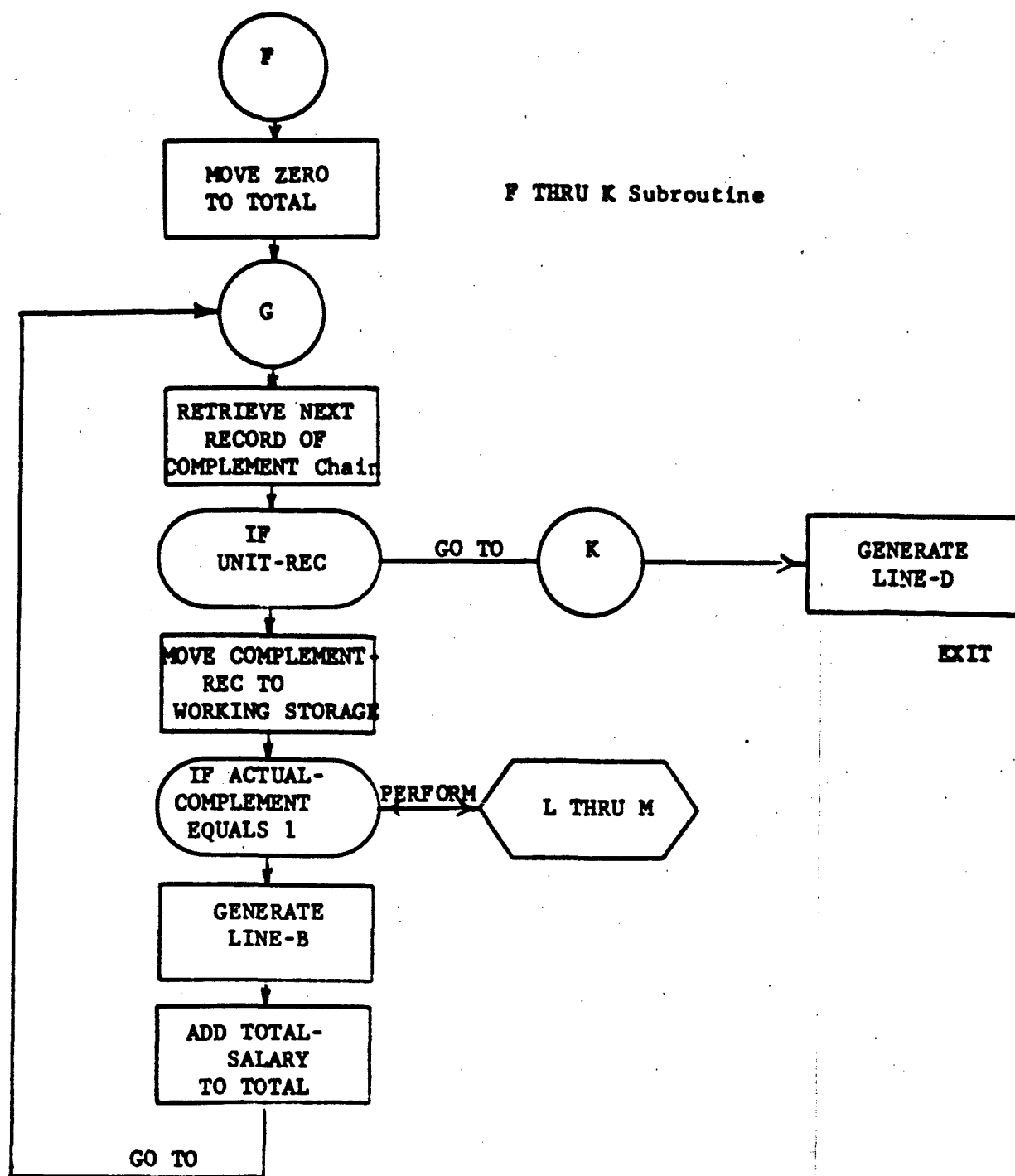


Figure 15. Flow Chart

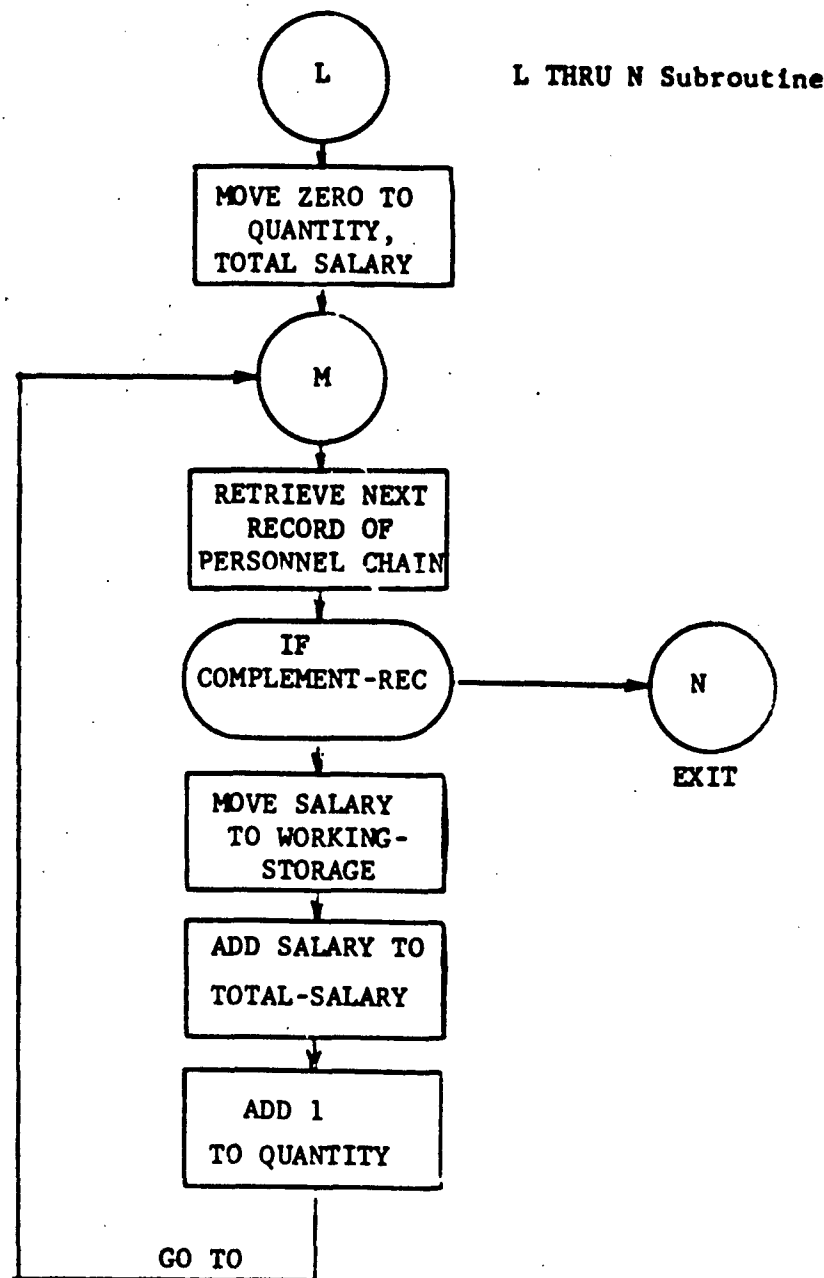


Figure 16. Flow Chart



Figure 17 is the program listing for report preparation routine shown by Figures 14, 15, and 16. Figure 18 shows the forms of the print lines referenced in the Budget Reports.

The Group Budget Report is very similar to the Section Budget Report. It adds additional data concerning lower level budgets that is new. Only the program listing is given for this report in Figure 19. The reader should note that the paragraphs H and I have been added to subroutine "F thru K" and the paragraphs produce LINE-C which contains the lower level section count and salary totals. If the reader refers back to Figure 9, he can see how the lower level UNIT-REC records could be accessed through the SUB-UNIT and ASSIGNMENT chains. The procedure is essentially the same used to take the hierarchical approach to unit sequencing in Figures 10 and 11.

The print lines which are required to print the specified reports are being produced through the use of the COBOL Report Writer to simplify the report coding. The same reports could have been produced through the use of the WRITE statement and normal COBOL output procedures. The REPORT SECTION has not been furnished, but the print line formats are shown in Figure 18.

The Exception Budget Report is specified in Attachment 9 of the case study. It is identical to the Section Budget Report except that only certain sections qualify.

Actual Number Employees > authorized number  
and [actual salaries > budget + \$800  
or actual salaries < budget - \$800]

Because of the similarities of these programs, only the program listing of the section selection logic will be presented. A program listing (Figure 20) would be inserted into the Section Budget Report immediately between the test to determine if the unit is a section, "IF SECTION-CODE IS ZERO GO TO Ø," and the command to GENERATE LINE-A (Figure 17).

The Hypothesis Budget Report (Attachment 10 of the case study) is also closely related to the Section and Group Budget Reports. In fact, the Hypothesis Report indicates that both sections and groups should be reported jointly and that the sections of each group should be reported immediately prior to the group. This sequence can readily be produced by introducing the report function logic at insertion point 2 in the hierarchical UNIT-REC record retrieval procedure shown in Figures 10 and 11. Figure 21 shows the program listing for the Hypothesis Report. The figure also includes a test of a field named COMPLEMENT-STATUS. This field carries three values: "A"--proposed addition; "D"--proposed deletion; and Blank--normal status.

All normal COMPLEMENT-REC records have a blank value in the COMPLEMENT-STATUS field.

NOTE SECTION BUDGET REPORT, INSERT AT POINT 0 OR 1.

IF SECTION-CODE IS ZERO GO TO 0.  
GENERATE LINE-A.  
MOVE ZERO TO ACTUAL-COMPLEMENT.  
PERFORM F THRU K.  
GENERATE LINE-E.  
MOVE TOTAL TO DEVIATION.  
MOVE 1 TO ACTUAL-COMPLEMENT.  
PERFORM F THRU K.  
SUBTRACT TOTAL FROM DEVIATION.  
GENERATE LINE-F.  
GO TO 0.

F.  
MOVE ZERO TO TOTAL.

G.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-REC GO TO K; ELSE  
\* MOVE TO WORKING-STORAGE.

IF ACTUAL-COMPLEMENT EQUALS 1 PERFORM L THRU N.  
GENERATE LINE-B.  
ADD TOTAL-SALARY TO TOTAL.  
GO TO G.

K.  
GENERATE LINE-D.

L.  
MOVE ZERO TO QUANTITY, TOTAL-SALARY.

M.  
\* RETRIEVE NEXT RECORD OF PERSONNEL CHAIN;  
\* IF COMPLEMENT-REC GO TO N; ELSE  
\* MOVE SALARY TO WORKING-STORAGE.  
ADD SALARY TO TOTAL-SALARY.  
ADD 1 TO QUANTITY.  
GO TO M.

N.  
EXIT.

O.

NOTE END OF REPORT FUNCTION.

\* IDS statements

Figure 17. Partial Program Listing

## NOTE BUDGET REPORT PRINT LINES.

## NOTE LINE-A IMAGE BELOW

ORG UNIT 9999  
 REPORTS TO 9999  
 ORG NAME XXXXXXXXXXXXXXXXXXXX

## AUTHORIZED COMPLEMENT

## NOTE LINE-B IMAGE BELOW

9999 XXXXXXXXXXXXXXXXXXXX ZZ,ZZZ,ZZ9

## NOTE LINE-C IMAGE BELOW

9999 SECTION ZZ,ZZZ,ZZ9

## NOTE LINE-D IMAGE BELOW

TOTAL ZZZ,ZZ9

## NOTE LINE-E IMAGE BELOW

## ACTUAL COMPLEMENT

## NOTE LINE-F IMAGE BELOW

DEVIATION FROM BUDGET ZZZ,ZZ9-

## NOTE LINE-G IMAGE BELOW

## PROPOSED COMPLEMENT

## NOTE SKILL REPORT PRINT LINES.

## NOTE PAGE HEADER IMAGE BELOW

NAME NUMBER UNIT SKILLS

## NOTE LINE-M IMAGE BELOW

XXXXXXXXXXXXXXXXXXXXXXXXX 99999 9999 9999 XXXXXXXXXXXXXXXXXXXX

## NOTE LINE-N IMAGE BELOW

9999 XXXXXXXXXXXXXXXXXXXX

Figure 18. Partial Program Listing

NOTE GROUP BUDGET REPORT, INSERT AT POINT 0 OR 1.

IF SECTION-CODE IS NOT ZERO GO TO 0.  
IF GROUP-CODE IS ZERO GO TO 0.  
GENERATE LINE-A.  
MOVE ZERO TO ACTUAL-COMPLEMENT.  
PERFORM F THRU K.  
MOVE TOTAL TO DEVIATION,  
MOVE 1 TO ACTUAL-COMPLEMENT.  
GENERATE LINE-E.  
PERFORM F THRU K.  
SUBTRACT TOTAL FROM DEVIATION.  
GENERATE LINE-F.  
GO TO 0.

F.  
MOVE ZERO TO TOTAL.

G.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-REC GO TO M; ELSE  
\* MOVE TO WORKING-STORAGE,  
IF ACTUAL-COMPLEMENT EQUALS 1 PERFORM L THRU N.  
GENERATE LINE-H.  
ADD TOTAL-SALARY TO TOTAL.  
GO TO G.

H.  
\* RETRIEVE NEXT RECORD OF SUB-UNIT CHAIN;  
\* IF UNIT-REC GO TO K,  
\* RETRIEVE MASTER RECORD OF ASSIGNMENT CHAIN;  
\* MOVE TO WORKING-STORAGE,  
MOVE ZERO TO SECTION QUANTITY, SECTION-TOTAL-SALARY.

I.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-REC GO TO J; ELSE  
\* MOVE TO WORKING-STORAGE,  
IF ACTUAL-COMPLEMENT EQUALS 1 PERFORM L THRU N.  
ADD QUANTITY TO SECTION-QUANTITY.  
ADD TOTAL-SALARY TO SECTION-TOTAL-SALARY.  
GO TO I.

J.  
\* GENERATE LINE-C.  
\* ADD SECTION-SALARY-TOTAL TO TOTAL.  
\* RETRIEVE CURRENT SUB-UNIT-REC.  
GO TO H.

K.  
GENERATE LINE-D.  
L.  
MOVE ZERO TO QUANTITY, TOTAL-SALARY.

M.  
\* RETRIEVE NEXT RECORD OF PERSONNEL CHAIN;  
\* IF COMPLEMENT-REC GO TO N; ELSE  
\* MOVE SALARY TO WORKING-STORAGE.  
\* ADD SALARY TO TOTAL-SALARY.  
\* ADD 1 TO QUANTITY.  
GO TO M.

N.  
EXIT.

O.

NOTE END OF REPORT FUNCTION.

Figure 19. Partial Program Listing

## NOTE EXCEPTION REPORT SELECTION FUNCTION.

IF SECTION-CODE IS ZERO GO TO O.  
MOVE ZERO TO BUDGET-SALARY, BUDGET-COUNT.

B.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-RECORD GO TO D; ELSE  
\* MOVE TO WORKING-STORAGE.  
IF COMPLEMENT-STATUS IS "A" GO TO B.  
ADD TOTAL-SALARY TO BUDGET-SALARY.  
ADD QUANTITY TO BUDGET-COUNT.

C.  
\* RETRIEVE NEXT RECORD OF PERSONNEL CHAIN;  
\* IF COMPLEMENT-REC GO TO B; ELSE  
\* MOVE SALARY TO WORKING-STORAGE.  
SUBTRACT SALARY FROM BUDGET-SALARY.  
SUBTRACT 1 FROM BUDGET-COUNT.  
GO TO C.

D.  
IF BUDGET-COUNT IS LESS THAN ZERO GO TO O.  
IF BUDGET-SALARY IS NOT LESS THAN 800 GO TO E.  
IF BUDGET-SALARY IS GREATER THAN -800 GO TO O.

E.

## NOTE END OF SELECTION FUNCTION.

Figure 20. Partial Program Listing

Before attempting to produce a Hypothesis Budget Report, all COMPLEMENT-REC records suggested for deletion would have their status changed to "D." All proposed additions would be stored with the status "A." Then when the report is run, the proposed deletions will not appear and the proposed additions will. These changes to the data base would be introduced by the data base maintenance routine to be discussed later.

## NOTE HYPOTHESIS BUDGET REPORT, INSERT AT POINT 2.

GENERATE LINE-A.  
MOVE ZERO TO ACTUAL-COMPLEMENT.  
PERFORM F THRU K.  
MOVE 1 TO ACTUAL-COMPLEMENT.  
GENERATE LINE-G.  
PERFORM F THRU K.  
GO TO O.

F.  
MOVE ZERO TO TOTAL.

G.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-REC GO TO M; ELSE  
\* MOVE TO WORKING-STORAGE.  
IF COMPLEMENT-STATUS IS "D" GO TO G.  
IF ACTUAL-COMPLEMENT EQUALS 1 PERFORM L THRU N.  
GENERATE LINE-H.  
ADD TOTAL-SALARY TO TOTAL.  
GO TO G.

H.  
\* RETRIEVE NEXT RECORD OF SUB-UNIT CHAIN;  
\* IF UNIT-REC GO TO K,  
\* RETRIEVE MASTER RECORD OF ASSIGNMENT CHAIN;  
\* MOVE TO WORKING-STORAGE.  
MOVE ZERO TO SECTION QUANTITY, SECTION-TOTAL-SALARY.

I.  
\* RETRIEVE NEXT RECORD OF COMPLEMENT CHAIN;  
\* IF UNIT-REC GO TO J; ELSE  
\* MOVE TO WORKING-STORAGE.  
IF ACTUAL-COMPLEMENT EQUALS 1 PERFORM L THRU N.  
ADD QUANTITY TO SECTION-QUANTITY.  
ADD TOTAL-SALARY TO SECTION-TOTAL-SALARY.  
GO TO I.

J.  
\* GENERATE LINE-C.  
\* ADD SECTION-TOTAL-SALARY TO TOTAL.  
\* RETRIEVE CURRENT SUB-UNIT-REC.  
GO TO H.

K.  
GENERATE LINE-D.

L.  
MOVE ZERO TO QUANTITY, TOTAL-SALARY.

M.  
\* RETRIEVE NEXT RECORD OF PERSONNEL CHAIN;  
\* IF COMPLEMENT-REC GO TO N; ELSE  
\* MOVE SALARY TO WORKING-STORAGE.  
ADD SALARY TO TOTAL-SALARY.  
ADD 1 TO QUANTITY.  
GO TO M.

N.  
EXIT.

O.

NOTE END OF REPORT FUNCTION.

Figure 21. Partial Program Listing

## SKILL REPORTS

The several skill reports specified (Attachment 9 of the case study) bear directly upon the information available concerning each employee. We know, for example, that each employee is classified with one primary skill and some number of secondary skills. The number of secondary skills is variable, from zero to many. Therefore, a chain will be established to associate SKILL-REC records with the PERSONNEL-REC record. Each SKILL-REC record represents a secondary skill. The SKILL chain links the SKILL-REC records to the PERSONNEL-REC record of the employee. It is also useful to group all employees who have the same skill for reporting purposes. These groupings would be very useful when the primary consideration of a personnel search is qualification in a specified skill or skills. A chain has been established to link together all of the records (PERSONNEL-REC and SKILL-REC) which represent the same value for SKILL-CODE. This chain has been named the SKILLEE chain. A special record type, SKILL-CLASS-REC record, has been established to serve as the master record of the SKILLEE chain and serves as an entrance point into the system. Figure 2 shows the data structure diagram for the entire data base. The reader can see how the SKILL-REC and SKILL-CLASS-REC records are integrated. Figure 30 at the end of the report contains the entire data description for the data base and corresponds to Figure 2.

The SKILL-CLASS-REC record is a calculated, or randomized, record and can be retrieved based upon the SKILL-CODE contained within, using the RETRIEVE statement.

### RETRIEVE data-name RECORD:

This statement will access the specific record of the datatype named, based upon the "RETRIEVAL VIA..." clause in the record definition, and the current data values for fields specified as randomize, sort and match fields.

If there is not a SKILL-CLASS-REC record containing the same SKILL-CODE value as the working storage field with the same name, the error exit will be taken. Otherwise, control will pass to the next command. Figure 22 is an example of the RETRIEVE verb used to retrieve a calculated record. The PERSONNEL-REC record, "RANDOMIZE EMPLOYEE-NO.," and the UNIT-REC record "RANDOMIZE UNIT-CODE" are also calculated records and provide an easy entrance into the data base from which chain-guided excursions may be made.

```
MOVE data-name TO SKILL-CODE
RETRIEVE SKILL-CLASS-REC
IF ERROR GO TO X
```

Figure 22. Program Listing

The UNIT-MASTER-REC has also been declared to be a calculated record, "RETRIEVAL VIA CALC CHAIN," however, it represents a special case. The page range statement has confined this record specifically to Page 1. No field has been specified as a randomized field. Therefore, the first UNIT-MASTER-REC stored on Page 1 will be accepted: all subsequent attempts to store additional UNIT-MASTER-REC records would automatically be rejected as duplicates and an error exit transfer made.

Figure 23 is a schematic diagram representing the records in the SKILLEE chain for the SKILL-CLASS-REC record containing the SKILL-CODE value of "3340." The records in this SKILLEE chain are the ones of interest when checking all personnel classified with the machine operator skill as either their primary skill or secondary skill.

The first person listed is Mr. Kent. "Machine Operator" is a secondary skill for him. His primary skill is "Product Engineer." Mr. Fletcher is the second person listed and "Machine Operator" is one of his secondary skills. Several more people are listed with secondary skills as Machine Operators.

Mr. Payne is the first person listed with "Machine Operator" as his primary skill.

In Figure 23, the SKILL-CLASS-REC record contains the SKILL-CODE written, "3340." This is to signify that the value is actually stored in the record. The PERSONNEL-REC record shows the same picture, "...," implying value contained. The data definitions for the SKILL-CLASS-REC and PERSONNEL-REC records contain 02 entries for the SKILL-CODE field (Figure 30). The SKILL-REC record is illustrated with the SKILL-CODE value written '(3340)'. This implies that the value exists implicitly for that record, but is not actually stored in it. There is no 02 entry for the SKILL-CODE field in the SKILL-REC record data definitions. When either the SKILL-CODE or SKILL-NAME that is implied by a particular SKILL-REC record is needed, the values may be retrieved from the SKILL-CLASS-REC record that is the SKILL-REC record's master. This is an example of eliminating redundant information. The "HEAD chain-name CHAIN" command will automatically retrieve that information. In the same way, the redundant POSITION-TITLE field has been removed from all of the PERSONNEL-REC records and is carried only in their master COMPLEMENT-REC record.

#### HEAD chain-name CHAIN:

This statement will access the master record of the current chain of the type named and automatically move the contents of its data fields to working storage. The master record does not become the current record of the named chain. However, it will become the current record of all chains in which the master record itself is a detail record. This permits HEAD statements to be executed at higher and higher levels of the data structure based upon the results of the HEAD statement.





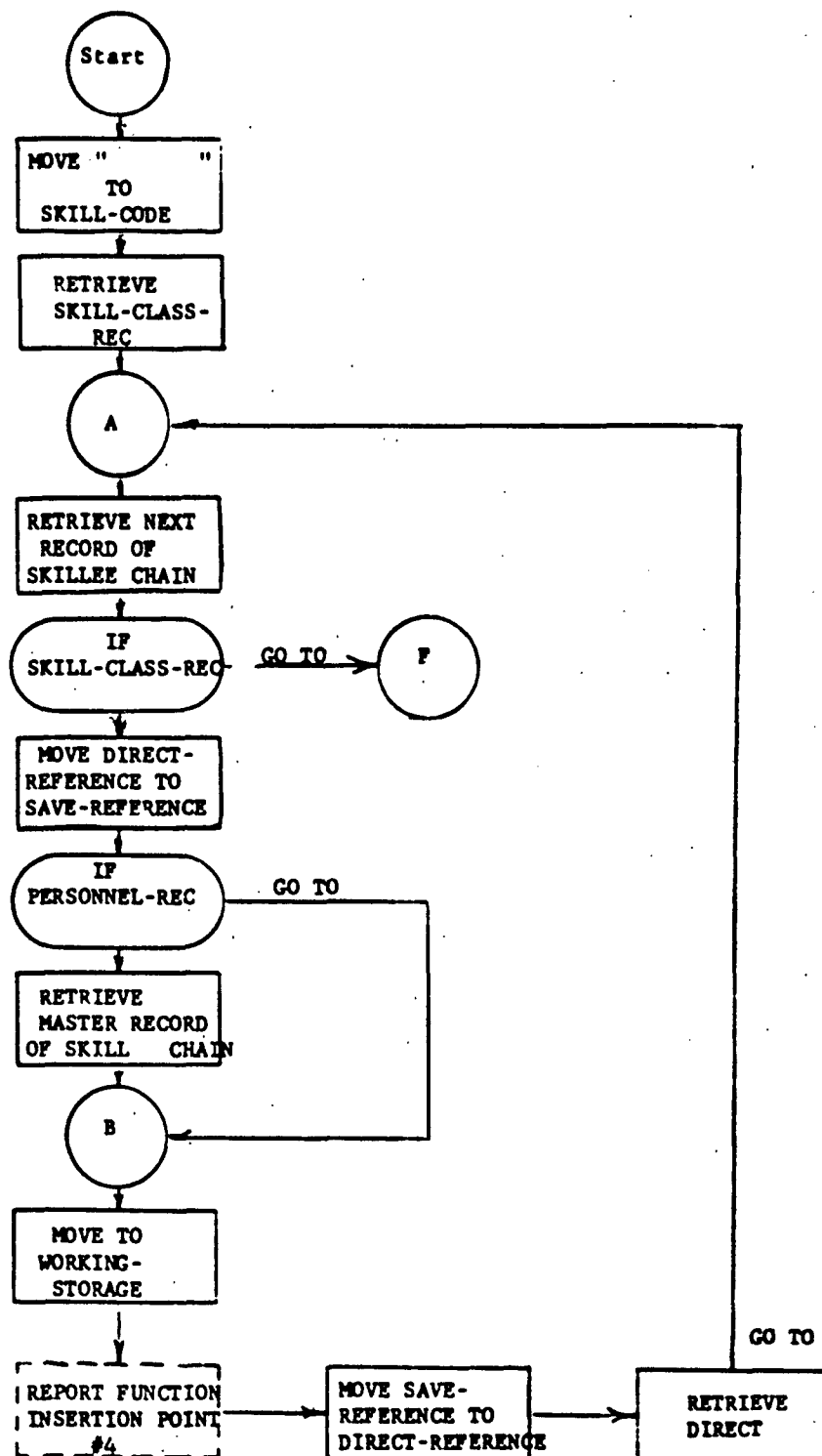


Figure 24. Flow Chart

The flow chart in Figure 24 illustrates a procedure which, given a value for SKILL-CODE, will sequentially retrieve the PERSONNEL-REC record for each person who is classified with that skill as either a primary or secondary skill.

The additional qualification of an employee to determine if he should be included on the report and the actual reporting coding would be inserted at the insertion point 4 that is indicated in the flow chart. Figure 25 is the IDS/ COBOL program listing that corresponds to the flow chart in Figure 24.

```
START.
*   OPEN DATA-BASE.
    OPEN OUTPUT REPORT-FILE.
    INITIATE ALL.
    MOVE 3340 TO SKILL-CODE,
*   RETRIEVE SKILL-CLASS-REC.
    A.
*   RETRIEVE NEXT RECORD OF SKILLEE CHAIN.
*   IF SKILL-CLASS-REC GO TO F.
    MOVE DIRECT-REFERENCE TO SAVE-REF.
    IF PERSONNEL-REC GO TO B.
*   RETRIEVE MASTER RECORD OF SKILL CHAIN.
    B.
*   MOVE TO WORKING-STORAGE,
    NOTE REPORT FUNCTION INSERTION POINT NUMBER 4.
    MOVE SAVE-REF TO DIRECT-REFERENCE.
*   RETRIEVE DIRECT.
    GO TO A.
    F.
*   TERMINATE ALL.
    CLOSE REPORT-FILE.
    CLOSE DATA-BASE.
    STOP RUN.
```

Figure 25. Partial Program Listing

RETRIEVE DIRECT:

This statement will access the record at the address equivalent to the value stored in the communication field name, "DIRECT-REFERENCE." The successful retrieval of a particular type of record may be tested using the "IF dataname RECORD GO TO statement-name" statement.

RETRIEVE DIRECT is a retrieval command that will allow you to drive directly to any place in the data base, just as long as you know the street name and house number. In the Integrated Data Store, the address is made up of a PAGE AND LINE number. After every IDS command, the address of the processed record may be found in the communication cell named "DIRECT-REFERENCE." This address may be saved for future use. Prior to using the RETRIEVE-DIRECT command to retrieve, the program must store an address value in DIRECT-REFERENCE.

The first demand report of Attachment 8 of the case study specifies a particular skill selection report:

Any skill code	= 3340
and total number of skills	> 2
and sex	= M
and birthdate	> 1916
and birthdate	< 1935

The COBOL coding to check all but the number of skills is obvious. The number of skills are quite easily determined by processing the records in the SKILL chain to determine the number of secondary skills. Figure 26 is the IDS/COBOL procedure to determine whether the employee, with the "Machine Operator" classification (3340), qualifies considering the other stated restrictions. If the employee is qualified, a print line, LINE-M, will be generated. A print line, LINE-N, will be generated for each of his secondary skills. Figure 26 is the program listing for Demand Report 1.

NOTE DEMAND REPORT NUMBER 1, INSERT AT POINT NUMBER 4.

```
IF LEVEL IS "HEAD" GO TO E.  
IF SEX IS NOT "M" GO TO E.  
IF YEAR IS NOT GREATER THAN 16 GO TO E.  
IF YEAR IS NOT LESS THAN 35 GO TO E.  
MOVE 1 TO COUNT.
```

C.

```
* RETRIEVE NEXT RECORD OF SKILL CHAIN:  
* IF PERSONNEL-REC GO TO E.  
* ADD 1 TO COUNT.  
* IF COUNT IS NOT GREATER THAN 2 GO TO C.  
* RETRIEVE CURRENT PERSONNEL-REC:  
* HEAD SKILLEE CHAIN.  
* GENERATE LINE-M.
```

D.

```
* RETRIEVE NEXT RECORD OF SKILL CHAIN  
* IF PERSONNEL-REC GO TO E; ELSE  
* HEAD SKILLEE CHAIN.  
* GENERATE LINE-N.  
* GO TO D.
```

E.

NOTE END OF REPORT FUNCTION.

Figure 26. Partial Program Listing

These lines are being generated in an "as occurs" basis. The assumption is that they will somehow be sorted in the output process into the employee name sequence to meet the report requirements. This was done to make the report extraction phase more clear to the reader. There are several practical means for sorting this report. One way would be to write a tape file, sort the tape records, and finally bring the records back for report formatting in report sequence. A second means would be to use the IDS facilities for sorting records, i.e., the sorted chain.

The second of the demand reports has been set up to print the reports in specified sequence. Three temporary record types, L-REC, M-REC, and N-REC have been defined, and a special working area established for their storage (PAGE RANGE 20001 to 21000). Figure 27 shows the data description for these three records. The L-REC serves as an overall control for the chain in which the M-REC records are sorted with the primary key EMPLOYEE-NAME and the secondary key EMPLOYEE-NO. The chain is specified, "DUPLICATES NOT ALLOWED" so that the same employee will not be reported twice if he is selected a second or third time under different criteria. If an M-REC record for a selected employee has already been stored, an attempt to store another will cause an error exit. In this case, the procedure will take the error exit to discontinue the processing of an employee who has already been processed.

Print line formats for the Skill Reports are shown at the bottom of Figure 18.

The program shown in Figure 28 will automatically print Skill Report 2 in sequence. Print line data is stored and sorted as M-REC and N-REC records in the Integrated Data Store until all of the data base has been processed. Then these records are retrieved, the data formatted for printing and then deleted. This process can be observed beginning at paragraph F in Figure 28. First the CURRENT and only L-REC record is retrieved.

Note that RETRIEVE CURRENT is a new retrieval verb and is equivalent to the statement "Go back to the last stop light." Then, starting at paragraph G, there is a RETRIEVE NEXT command that will, when used repeatedly, pull out all of the M-REC records in the EMPLOYEE-NAME sequence. The M-REC records contain employee information. The M-REC records are "MOVE TO WORKING-STORAGE" so that the COBOL report writer can access the data fields on "GENERATE LINE-M." Then the M-REC record is deleted. When a master record is deleted (M-REC is the master of N-REC), all the detail records are automatically deleted. The IDS language permits the DELETE command to be conditioned so that the deletion of a detail is recognized. The following statement, ON N-REC DETAIL; MOVE TO WORKING-STORAGE; PERFORM H, will cause IDS to catch each N-REC record in the deletion process and unpack its data fields into working storage. Then it calls for the performance of the COBOL-written paragraph H. Paragraph H is a call on the COBOL Report Writer; "GENERATE LINE-N" specifies generation of the secondary skill print line. When an M-REC record deletion is complete (all N-REC records deleted and formatted for printing), then the coding status

01 L-REC;  
    TYPE IS 100;  
    RETRIEVAL VIA L-CODE;  
    PAGE-RANGE IS 20001 TO 20001.  
02 L-CODE; SIZE 8 NUMERIC.  
98 M CHAIN MASTER;  
    CHAIN-ORDER IS SORTED.

01 M-REC;  
    TYPE IS 101;  
    RETRIEVAL VIA M CHAIN;  
    PAGE-RANGE IS 20001 TO 21000.  
02 EMPLOYEE-NAME; SIZE 25 ALPHANUMERIC.  
02 EMPLOYEE-NO; SIZE 5 NUMERIC.  
02 SKILL-CODE; SIZE 4 NUMERIC.  
02 SKILL-NAME; SIZE 20 ALPHANUMERIC.  
02 UNIT-CODE, SIZE 4 NUMERIC.  
98 M CHAIN DETAIL;  
    SELECT CURRENT MASTER;  
    ASCENDING KEY IS EMPLOYEE-NAME;  
    ASCENDING KEY IS EMPLOYEE-NO;  
    DUPLICATES NOT ALLOWED.  
98 N CHAIN MASTER;  
    CHAIN-ORDER IS AFTER.

01 N-REC  
    TYPE IS 102;  
    RETRIEVAL VIA N CHAIN;  
    PAGE-RANGE IS 20001 TO 21000.  
02 SKILL-CODE; SIZE 4 NUMERIC.  
02 SKILL-NAME; SIZE 20 ALPHANUMERIC.  
98 N CHAIN DETAIL;  
    SELECT CURRENT MASTER.

Figure 27. Partial Data Description Listing

```
PROGRAM-ID. DEMAND-REPORT-2.  
START.  
*   OPEN DATA-BASE.  
    OPEN OUTPUT REPORT-FILE.  
    INITIATE ALL.  
    STORE L-REC.  
*   MOVE 111U TO SKILL CODE  
    RETRIEVE SKILL-CLASS-REC.  
    MOVE 112V TO CURRENT-CODE.  
    GO TO A.  
AA.  
    ADD 1 TO CURRENT-CODE.  
    MOVE CURRENT-CODE TO SKILL-CODE.  
    IF SKILL-CODE IS EQUAL 114U GO TO F.  
    RETRIEVE SKILL-CLASS-REC.  
    IF ERROR GO TO AA,  
*   A.  
    RETRIEVE NEXT RECORD OF SKILLEE CHAIN.  
    IF SKILL-CLASS-REC GO TO AA.  
    MOVE DIRECT-REFERENCE TO SAVE-REF.  
    IF PERSONNEL-REC GO TO B.  
    RETRIEVE MASTER RECORD OF SKILL CHAIN.  
*   B.  
    MOVE TO WORKING-STORAGE.  
    IF LEVEL IS "HEAD" GO TO E.  
    IF SEX IS NOT "M" GO TO E.  
    HEAD SKILLEE CHAIN.  
    STORE M-REC.  
    IF ERROR GO TO E.  
*   D.  
    RETRIEVE NEXT RECORD OF SKILL CHAIN  
    IF PERSONNEL-REC GO TO E; ELSE  
    HEAD SKILLEE CHAIN,  
    STORE N-REC.  
    GO TO D.  
*   E.  
    MOVE SAVE-REF TO DIRECT-REFERENCE.  
    RETRIEVE DIRECT.  
    GO TO A.  
*   F.  
    RETRIEVE CURRENT L-REC.  
*   G.  
    RETRIEVE NEXT RECORD OF M CHAIN;  
    IF L-REC DELETE; GO TO I; ELSE  
    IF M-REC MOVE TO WORKING-STORAGE.  
    GENERATE LINE-M.  
    DELETE;  
    ON N-REC DETAIL MOVE TO WORKING-STORAGE; PERFORM M.  
    GO TO G.  
*   H.  
    GENERATE LINE-N.  
*   I.  
    TERMINATE ALL.  
    CLOSE REPORT-FILE.  
    CLOSE DATA-BASE.  
    STOP RUN.
```

Figure 28. Partial Program Listing

simply states "GO TO G," where a new M-REC will be retrieved. This process will continue until all the M-REC records are deleted, and the L-REC is retrieved again after circumnavigating the chain. Then the L-REC record is deleted and control is transferred to the end of the program.

The use of sorted chains for report preparation is very useful. However, the reader should note that this use should be approached with the view of overall system economy in mind. The Integrated Data Store sorts records (chains) by the insertion process, the only way to maintain a chain in sequence at all times. Each time a new record is stored, it is linked into the chain in its proper position.

Merge sorts are much more efficient when there is a large number of records to be sorted at a given time. Therefore, long sorted reports probably should be sorted off line by a tape or disc sort. Long sorted chains for master file purposes can be streamlined through the use of intermediate level master records which effectively cut down the length of the chain and materially improve the efficiency of the insertion process:

This technique would have been applied to the UNIT chain if there were to have been a large number of UNIT-REC records. Figure 29 illustrates the Data Structure Diagram of such an approach.

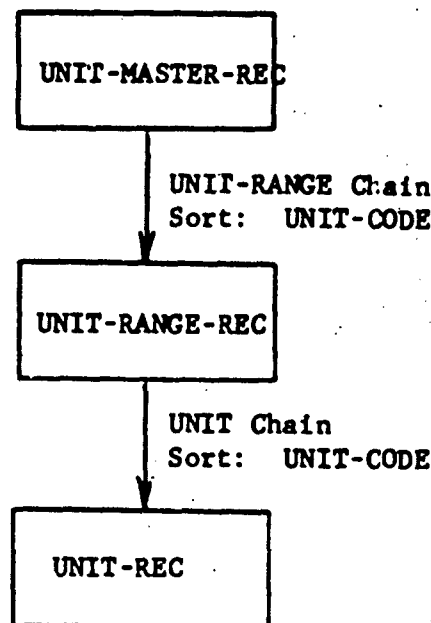


Figure 29. Data Structure Diagram



For an example, if two thousand UNIT-REC records were planned for the system, then approximately one hundred UNIT-RANGE-REC records would be stored with UNIT-CODE values which are designed so that the difference between two consecutive UNIT-CODE values would cover about one percent of the actual UNIT-CODE value series. Then the UNIT-REC records would be sequenced under their appropriate UNIT-RANGE-REC record.

If the UNIT-RANGE-REC records were not used, the storage of a new UNIT-REC would have required the searching of part or perhaps all of the UNIT chain before a new UNIT-REC record found its proper position in the chain. Randomly selected SKILL-CODE values would, on the average, require searching of half the chain or 1000 record retrievals in the example cited. If UNIT-RANGE-REC records were used, then storage would require searching halfway through the UNIT-RANGE-REC records (50 records) and halfway through UNIT-REC records (10 records) attached to the UNIT-RANGE-REC. The system could be arranged so that all of the UNIT-RANGE-REC records were stored in the same page as the UNIT-MASTER-REC record.

Pages have been mentioned several places in the text. A page is a data block of a few hundred to a few thousand characters (user option) in which logical records are stored. Every transmission between core and disc represents a page transfer. When records of a chain are stored in the same page, access time per record retrieved is greatly reduced. The storage of records may be controlled to facilitate clustering of details within a page.

Under these circumstances, all one hundred UNIT-RANGE-REC records could be retrieved with one seek/read of the disc file; that portion of the search would be extremely fast. One thousand seek/read would be reduced to eleven, and execution time would be reduced from a minute and a half to about one second.

Interestingly, the unit hierarchy structure with its intermediate SUB-UNIT-REC record serves a very similar function by splitting the records into smaller chains to be sequenced. As the previous text indicated, the UNIT-MASTER-REC record and its UNIT chain are not really needed because the unit hierarchical chains can serve the same purpose. However, the chains did represent the easiest way to begin the explanation and were introduced for pedagogical reasons. They probably would not need to exist in a real system as long as the UNIT-CODE is functionally assigned by unit hierarchy.

The need to periodically produce reports in a particular sequence is a legitimate reason for establishing permanently sorted chains in the data base--once the chains are sorted, they are always sorted. Then data can be extracted in report sequence by simply steering the retrieval process along the specified chain.

## FILE MAINTENANCE

All of the aspects of the information content and the chain relations have been discussed for the data base problem. The detailed data description is listed in Figure 30. The IDS SECTION must immediately follow the COBOL WORKING-STORAGE SECTION. All of the IDS record data fields will appear as WORKING-STORAGE fields. All COBOL field processing statements, MOVE, ADD, IF, etc., will access the fields as found in WORKING STORAGE.

A file maintenance program that is exceedingly simple has been prepared. Basically, it reads card images off an input file and branches in accordance to a card code. There is a unique code established for each type of data base record except the UNIT-MASTER-REC record which will be stored only once. The input card for a particular data base record contains values for all of the datafields necessary to store a record. This includes all of the 02 fields plus any MATCH-KEY fields which are not actually stored in the record.

The input card also contains a control field named "C-STATUS." This control field will contain the following values:

"S"	Store
"D"	Delete
"M"	Modify

The file maintenance routine will carry out the function indicated by the control field value. If the function indicated is STORE or MODIFY, the entire card must be punched with the desired values as they will be after file maintenance is complete. If the function indicated is DELETE, only the MATCH-KEYs and sort keys, or RANDOMIZE fields specified for the retrieval phase, need be punched. As an example, the PERSONNEL-REC record is specified "RETRIEVAL VIA CALC CHAIN." Under the 98 entry for the CALC chain the phrase "RANDOMIZE EMPLOYEE-NO" exists.

A card punch:

04	CARD-CODE
D	C-STATUS
33144	C-EMPLOYEE-NO

would be adequate to carry out a retrieval and deletion of the personnel record for Quinn, S.M., as specified in Attachment 6 of the case study. It would also delete all of his SKILL-REC records.

The addition of a new SKILL-REC record would be punched:

05	CARD-CODE
S	C-STATUS
85657	C-EMPLOYEE-NO
3570	C-SKILL-CODE

## IDS SECTION.

MD DATA-BASE; PAGE CONTAINS 1920 CHARACTERS;  
FILE CONTAINS 100000 PAGES.

## 01 UNIT-MASTER-REC;

TYPE IS 070;

RETR' VAL VIA CALC CHAIN;

PAGE-RANGE 1 TO 1.

98 CALC CHAIN DETAIL;

98 UNIT CHAIN MASTER;

CHAIN-ORDER IS SORTED.

## 01 UNIT-REC;

TYPE IS 010;

RETRIEVAL VIA CALC CHAIN;

PAGE-RANGE IS 1 TO 20000.

02 UNIT-CODE; SIZE 4 NUMERIC.

03 DIVISION-CODE; SIZE 1 NUMERIC.

03 DEPARTMENT-CODE; SIZE 1 NUMERIC.

03 GROUP-CODE; SIZE 1 NUMERIC.

03 SECTION-CODE; SIZE 1 NUMERIC.

02 REPORTING-UNIT; SIZE 4 NUMERIC.

02 ORG-NAME; SIZE 20 ALPHANUMERIC.

02 TOTAL-BUDGET; SIZE 7 NUMERIC.

98 CALC CHAIN DETAIL;

RANDOMIZE UNIT-CODE.

98 SUB-UNIT CHAIN MASTER;

CHAIN-ORDER IS SORTED.

98 ASSIGNMENT CHAIN MASTER;

CHAIN-ORDER IS FIRST.

98 COMPLEMENT CHAIN MASTER;

CHAIN-ORDER IS SORTED.

98 UNIT CHAIN DETAIL;

SELECT CURRENT MASTER;

ASCENDING KEY IS UNIT-CODE;

DUPLICATES NOT ALLOWED.

## 01 SUB-UNIT-REC;

TYPE IS 030;

RETRIEVAL VIA SUB-UNIT CHAIN;

PAGE-RANGE IS 1 TO 20000.

02 SUB-UNIT-CODE; SIZE 4 NUMERIC.

98 SUB-UNIT CHAIN DETAIL;

SELECT UNIQUE MASTER;

MATCH-KEY IS UNIT-CODE;

ASCENDING KEY IS SUB-UNIT-CODE;

DUPLICATES NOT ALLOWED.

98 ASSIGNMENT CHAIN DETAIL;

SELECT UNIQUE MASTER;

MATCH-KEY IS SUB-UNIT-CODE SYNONYM UNIT-CODE.

Figure 30. Partial Data Description Listing  
(Sheet 1 of 3)

## 01 COMPLEMENT-REC;

TYPE IS 040;

RETRIEVAL VIA COMPLEMENT CHAIN;

PAGE-RANGE IS 1 TO 20000.

02 COMPLEMENT-STATUS; SIZE 1 ALPHANUMERIC.

02 JOB-CODE; SIZE 4 NUMERIC.

02 POSITION-TITLE; SIZE 20 ALPHANUMERIC.

02 QUANTITY; SIZE 2 NUMERIC.

02 TOTAL-SALARY; SIZE 6 NUMERIC.

98 COMPLEMENT CHAIN DETAIL;

SELECT UNIQUE MASTER;

MATCH-KEY IS UNIT-CODE;

ASCENDING KEY IS JOB-CODE.

DUPLICATES NOT ALLOWED;

LINKED TO MASTER.

98 PERSONNEL CHAIN MASTER;

CHAIN-ORDER LAST.

## 01 PERSONNEL-REC;

TYPE IS 050;

RETRIEVAL VIA CALC CHAIN;

PAGE-RANGE IS 1 TO 20000.

02 EMPLOYEE-NAME; SIZE 25 ALPHANUMERIC.

02 EMPLOYEE-NO; SIZE 5 NUMERIC.

02 UNIT-CODE, SIZE 4 NUMERIC.

02 JOB-CODE, SIZE 4 NUMERIC.

02 LEVEL; SIZE 4 ALPHANUMERIC.

02 SALARY; SIZE 6 NUMERIC.

02 SKILL-CODE; SIZE 4 NUMERIC.

02 SEX; SIZE 1 ALPHANUMERIC.

02 BIRTHDATE; SIZE 6 ALPHANUMERIC.

03 MONTH; SIZE 2 ALPHANUMERIC.

03 DAY; SIZE 2 ALPHANUMERIC.

03 YEAR; SIZE 2 ALPHANUMERIC.

98 CALC CHAIN DETAIL;

RANDOMIZE EMPLOYEE-NO.

98 PERSONNEL CHAIN DETAIL;

SELECT UNIQUE MASTER;

MATCH-KEY IS UNIT-CODE;

MATCH-KEY IS JOB-CODE;

LINKED TO MASTER.

98 SKILL CHAIN DETAIL;

SELECT UNIQUE MASTER;

MATCH-KEY IS SKILL-CODE;

LINKED TO MASTER.

98 SKILL CHAIN MASTER;

CHAIN-ORDER IS FIRST.

Figure 30. Partial Description Listing  
(Sheet 2 of 3)

01 SKILL-CLASS-REC;  
TYPE IS 020;  
RETRIEVAL VIA CALC CHAIN;  
PAGE-RANGE IS 1 TO 20000.  
02 SKILL-NAME; SIZE 20 ALPHANUMERIC.  
02 SKILL-CODE; SIZE 4 NUMERIC.  
98 CALC CHAIN DETAIL;  
RANDOMIZE SKILL-CODE.  
98 SKILLEE CHAIN MASTER;  
CHAIN-ORDER FIRST.

01 SKILL-REC;  
TYPE IS 060;  
RETRIEVAL VIA SKILL CHAIN;  
PAGE-RANGE IS 1 TO 20000.  
98 SKILL CHAIN DETAIL;  
SELECT UNIQUE MASTER;  
MATCH-KEY IS EMPLOYEE-NO.  
98 SKILLEE CHAIN DETAIL;  
SELECT UNIQUE MASTER;  
MATCH-KEY IS SKILL-CODE;

Figure 30. Partial Data Description Listing  
(Sheet 3 of 3)

This card would cause a new SKILL-REC record to be stored and linked to the appropriate PERSONNEL-REC and SKILL-CLASS-REC records

A PERSONNEL-REC record would be retrieved and modified if the following information came in on an input card:

04	CARD-CODE
M	C-STATUS
91152	C-EMPLOYEE-NO
GARBER, B. E.	C-EMPLOYEE-NAME
2111	C-UNIT-CODE
1330	C-JOB-CODE
EMPL	C-LEVEL
8200	C-SALARY
1330	C-SKILL-CODE
M	C-SEX
070734	C-BIRTHDATE
91152	NEW-EMPLOYEE-NO

Figure 31 is the data definition of the input card required to store, modify, or delete a PERSONNEL-REC.

```

01 EMPLOYEE-CARD.
  02 CARD-CODE; SIZE 2 NUMERIC.
  02 C-STATUS; SIZE 1 ALPHANUMERIC.
  02 C-EMPLOYEE-NO; SIZE 5 NUMERIC.
  02 C-EMPLOYEE-NAME; SIZE 25 NUMERIC.
  02 C-UNIT-CODE; SIZE 4 NUMERIC.
  02 C-JOB-CODE; SIZE 4 NUMERIC.
  02 C-LEVEL; SIZE 4 ALPHANUMERIC.
  02 C-SALARY; SIZE 6 NUMERIC.
  02 C-SKILL-CODE; SIZE 4 NUMERIC.
  02 C-SEX; SIZE 1 ALPHANUMERIC.
  02 C-BIRTHDATE; SIZE 6 NUMERIC.
  02 NEW-EMPLOYEE-NO; SIZE 5 NUMERIC.

```

Figure 31. Data Description

The portion of the file maintenance routine to process a PERSONNEL-REC change is listed in Figure 32. It is shown with the beginning and run-out portion of the maintenance program.

PROGRAM-ID. FILE MAINTENANCE.  
START.  
\* OPEN DATA-BASE.  
\* OPEN INPUT CARD-FILE.  
A.  
READ CARD-FILE; AT END GO TO K.  
GO TO B, C, D, E, F, & DEPENDING ON CARD-CODE.  
GO TO A.  
G.  
IF C-STATUS EQUALS "D" GO TO J.  
MOVE C-EMPLOYEE-NO TO EMPLOYEE-NO.  
MOVE C-EMPLOYEE-NAME TO EMPLOYEE-NAME.  
MOVE C-UNIT-CODE TO UNIT CODE.  
MOVE C-JOB-CODE TO JOB CODE.  
MOVE C-LEVEL TO LEVEL.  
MOVE C-SALARY TO SALARY.  
MOVE C-SKILL-CODE TO SKILL-CODE.  
MOVE C-SEX TO SEX.  
MOVE C-BIRTHDATE TO BIRTHDATE.  
IF C-STATUS EQUALS "S" GO TO H.  
IF C-STATUS EQUALS "M" GO TO I.  
GO TO A.  
H.  
\* STORE PERSONNEL-REC.  
\* GO TO A.  
I.  
\* RETRIEVE PERSONNEL-REC;  
\* IF ERROR GO TO A.  
\* MOVE NEW-EMPLOYEE-NO TO EMPLOYEE-NO.  
\* MODIFY EMPLOYEE-NO, EMPLOYEE-NAME, UNIT-CODE,  
\* JOB-CODE, LEVEL, SALARY, SKILL-CODE, SEX,  
\* BIRTHDATE.  
GO TO A.  
J.  
\* MOVE C-EMPLOYEE-NO TO EMPLOYEE-NO.  
\* RETRIEVE PERSONNEL-REC;  
\* IF ERROR GO TO A; ELSE DELETE.  
GO TO A.  
K.  
\* CLOSE CARD-FILE.  
\* CLOSE DATA-BASE.  
\* STOP RUN.

Figure 32. Program Listing

The modification portion of this file maintenance procedure has been written in a very simplified manner. We are forcing the user to input all the data field values for a record even if only one is being modified. The procedure also tells the Integrated Data Store to modify all of the data fields. Actually, IDS will only modify those fields where the card input values are different than those contained within the actual record. The reader should also note that the input card contains two employee-number fields: C-EMPLOYEE-NO. and NEW-EMPLOYEE-NO. The first is necessary for the retrieval process. The second represents the value of EMPLOYEE-NO. that is to be in the record when the modification is complete. Therefore, two such fields are required, the before and the after. If there is to be no change, they should contain the same value. It is very important to realize that all data fields associated with a record are subject to modification. This modification can include fields specified for randomization as well as match and sort fields. The system will automatically carry out the rerandomizing, resequencing, and reselection of master records where appropriate. The Integrated Data Store is the only system known to the author which automatically handles the modification of sort control fields, match fields, or randomize fields.

#### CONTROL REPORT

The control report program gives the user a chance to see another retrieval verb in action.

##### RETRIEVE EACH: AT END GO TO Procedure-name:

This statement will access the record at the address equivalent to the value stored in the communication cell named "FIRST-REFERENCE," or the first record with an address greater than the value stored. If the record retrieved has an address greater than the value stored in the communication cell named "LAST-REFERENCE," control will be transferred to the procedure named. If an acceptable record is retrieved, its address, incremented by one, will be stored in FIRST-REFERENCE so that the statement may be used again in a loop procedure without the programmer needing to update FIRST-REFERENCE. (See Figure 33.)

The RETRIEVE EACH command is very useful to the programmer when there is not a chain structure to guide his retrieval and he needs to retrieve all records of some type.

The entire Control Report coding is very short, and the control totals have been programmed for display on the console typewriter.

#### SUMMARY

This presentation has provided solutions to all of the questions raised by the case study. All reporting and file maintenance problems have been addressed. Although the programs supplied have not been compiled and tested, they represent coding for a real system that is in daily operation several places in the United States and whose application is growing rapidly.



PROGRAM-ID. CONTROL-REPORT.  
START.  
\* MOVE ZERO TO SALARY-COUNT, PERS-COUNT, UNIT-COUNT.  
OPEN DATA-BASE.  
MOVE 00000065 TO FIRST-REFERENCE.  
MOVE 01280000 TO LAST-REFERENCE.  
A.  
\* RETRIEVE EACH; AT END GO TO D.  
\* IF UNIT-REC GO TO B; ELSE  
\* IF PERSONNEL-REC MOVE SALARY TO WORKING-STORAGE;  
\* GO TO C.  
GO TO A.  
B.  
ADD 1 TO UNIT-COUNT.  
GO TO A.  
C.  
ADD SALARY TO SALARY-COUNT.  
ADD 1 TO PERS-COUNT.  
GO TO A.  
D.  
NOTE CONTROL-REPORT PRODUCED ON CONSOLE TYPWRITER.  
DISPLAY "NUMBER OF PERSONS EMPLOYED ", PERS-COUNT.  
DISPLAY "TOTAL ACTUAL ANNUAL SALARIES ", SALARY-COUNT.  
DISPLAY "NUMBER OF ORGANIZATION UNITS ", UNIT-COUNT.  
\* CLOSE DATA-BASE.  
STOP RUN.

Figure 33. Partial Program Listing

The case problem has been treated as a single integrated file with all meaningful relationship established through IDS's chaining capabilities. The data involved can be processed quickly even if it were only a very small portion of a large data base. The records themselves have been stripped of most of their redundant data fields to condense the file size. Figure 34 shows the record types, their character size, and record count. Record sizes and counts have been extended, and totals recorded by record type for the entire file.

<u>Record Name</u>	<u>Character Size</u>	<u>Record Count</u>	<u>Total Characters</u>
UNIT-MASTER-REC	13	1	13
UNIT-REC	60	30	1,800
SUB-UNIT-REC	21	29	609
COMPLEMENT-REC	54	77	4,158
PERSONNEL-REC	83	104	8,632
SKILL-CLASS-REC	37	47	1,739
SKILL-REC	17	100	1,700
AVERAGE RECORD SIZE	49		
	Total	388	18,651

Figure 34. Data Space Utilization

#### REFERENCES

1. "Introduction to Integrated Data Store," General Electric Computer Department Bulletin CPB-1048.
2. C. W. Bachman and S. B. Williams, "A General Purpose Programming System for Random Access Memories," AFIPS Fall Joint Computer Conference, 1964.
3. C. W. Bachman, "Software for Random Access Processing," Datamation, April 1965.
4. C. W. Bachman, "Integrated Data Store," DPMA Quarterly, January 1965.
5. R. G. Canning, "New Approaches to Random Access Files," EDP Analyzer, May 1965.
6. Kavanagh, T. F., "TABSOL - A Fundamental Concept for Systems Oriented Languages," Eastern Joint Computer Conference, December 1960.
7. McGee, W. C., "Generalization - Key to Successful Electronic Data Processing," Journal of the ACM, January 1959.

SECTION IVDESCRIPTION OF SDC DATA BASE SYSTEMS  
DEMONSTRATED AT SYMPOSIUM

This section contains a brief description of the three SDC data base systems that were demonstrated during the symposium. LUCID and General Purpose Display System are experimental systems developed by SDC under ARPA sponsorship. The ECCO/EPIC systems are operational systems for storage and retrieval of SDC personnel data. All three systems operate on-line as part of the SDC/ARPA time-sharing system on the Q-32 computer.

LUCID.....	4-3
General Purpose Display System.....	4-7
ECCO and EPIC.....	4-9

LUCID\*

Robert E. Bleier  
TSS-LUCID Project Leader  
System Development Corporation

LUCID is a general, computer-based data-management system that organizes, stores, and updates large, complex bodies of data. It also enables the user to retrieve information selectively from the organized data.

LUCID is general-purpose in that it can manipulate a large variety of data; it is special-purpose in that it operates only on the AN/FSQ-32 computer and requires the SDC Time-Sharing System to support its operation.

LUCID allows a user, relatively unsophisticated in computer programming technology, to develop rapidly his own data-management system without learning a complex programming language or having to consult an expert. Users who have not already structured their raw input data have used LUCID and have been running their data-management systems in one week. Some users who have already structured their raw input data may have adopted forms that must be preprocessed before LUCID can handle the data.

To construct a LUCID data base one must perform three major tasks: (1) describe the data that will be contained in the data base; (2) describe the format in which to present the data; and (3) present the actual data. Although the above sequence is logically desirable, it is not essential that the user follow this order. The products of tasks 1 and 2, the "Data Description" and the "Symbolic-Input Description," respectively, are translated by the system into a "User-Master" tape. The product of step 3, the "Data Entries," is translated into a "Data Base" tape.

Data base construction requires that a user be quite familiar with his data and have some general notion of the uses to which they will be put. LUCID does not eliminate the need for planning.

The system operates in the following way. The output of tasks 1 and 2 is input to the LUCID Translator. The Translator generates a dictionary table from the user's description of data items, substitutes a 24-bit code (called an OPAQUE tag) for all literal names and values, constructs tables of these codes along with their original literal value so that future tasks in the system may translate backwards or forwards through the codes, and produces the "User-Master" tape, containing the dictionary, tag conversion tables, and card format tables. The substitution of 24-bit code for literal values that tend to be of unknown length and somewhat repetitive saves a good deal of space and simplifies the

---

\*The development of this system is supported by the Advanced Research Projects Agency, under contract SD-97.

structure of the data base entries themselves. A slight time penalty is paid when loading and retrieving these literals because of the extra table lookup and conversion required.

The output of task 3 and of the Translator are input to the LUCID Loader. The Loader then converts all of the raw data according to the dictionaries supplied by the Translator, adds to the table of OPAQUE tags all new literals, and makes up the data base entries.

In order to facilitate retrieval, most data base load systems allow the user to specify how he wants his data organized. At load time this information is not always known, and if it is known, sometimes several different organizations are desirable. The LUCID Loader obviates this problem by constructing a retrieval tree structure. At the top of the tree is a table (COIL) containing the OPAQUE tag name of each item that actually has a value in the data base. Associated with each item tag is a pointer to another table (JUST) which contains a list of the unique values that occur for each item. Associated with each value in JUST is another pointer to a table (ALTO) containing the entry list where all occurrences of a value may be found in the data base. The entire group of tables is called the Concordance. For retrieval, the data base may be said to be totally organized in that any item may be used as a retrieval key. Retrieval using such a mechanism is rapid, providing that the Concordance fits within fast random-access auxiliary storage, such as drums or disc.

When the Loader has completed its operation, the user has a tape containing all the tables and data required for querying.

The LUCID query system that uses the Loader's output is currently being converted to time-sharing operation.

One of the systems currently operating under time-sharing [the General Purpose Display System (GPDS)--see p. 4-7. requires LUCID-produced data bases organized in the conventional fashion (entries within tables and tables within files). The program that does this reorganizing is called GOR.

GOR's design was influenced by the following factors: Disc would be the auxiliary storage medium for GPDS data bases, and any particular process built with GPDS would reference a fixed set of key items. Therefore, a mechanism is required that allows users to organize LUCID-produced data bases based on these key items. The organization had to minimize the number of disc transfers required in querying that data base. This is of no small importance as the average access time for the Q-32 disc is approximately 2/10 of a second per 4096 words (4096 words equaling a track).

GOR, therefore, was designed to organize entries in LUCID data bases into tables and organize tables into files. The user interacts with GOR on-line to specify the items whose values in the data base are to be used as the criteria for making these groupings. As many different organizations may be produced from a single LUCID data base as a user finds necessary. Depending upon the size of these data bases and the load on the time-sharing system, new data base tape can be generated by GOR in as little as ten minutes.

In the process of testing GOR and the data base retrieval components within GPDS, it became desirable to develop an on-line query program. This program, called QUIZ, has a language adapted from the LUCID query system and represents the first LUCID-time-sharing retrieval capability. With minor exceptions, QUIZ is composed of the same data base retrieval routines used in GPDS. QUIZ prints, counts or computes item values unconditionally or conditionally.

One of the data bases with which we are experimenting contains information about SDC divisions, customers and contracts. A QUIZ of this data base might look like

PRINT PROJECT AND AMOUNT CURRENT WHERE CUSTOMER EQ USAF AND DIVISION EQ DSD.

The data base is organized by CUSTOMER and SDC DIVISION. Therefore, before any data base searching is performed to generate the answers to this query, the values USAF and DSD are used to limit the range of entries within which QUIZ can look for PROJECT and AMOUNT CURRENT.

Because the data base is organized by CUSTOMER and SDC DIVISION, all the entries equal to USAF and DSD are together and probably reside within one disc access worth of data. Retrieval in this case will be instantaneous.

If the data base were not organized by CUSTOMER CLASS and DIVISION, the entire data base would have to be scanned to answer the query. This would entail many disc transfers and much computing.

Indications from users are that the QUIZ language is easy to learn and that the system is quite responsive.

The following operators (shown in capital letters) indicate the QUIZ language:

	EXISTS	
	FAILS	
	or	
PRINT or COUNT items or entries WHERE item	EQUAL TO	} value AND OR etc.
	NOT EQUAL TO	
	LESS THAN	
	LESS OR EQUAL	
	GREATER THAN	
	GREATER OR EQUAL	

(In the program, the value operators are abbreviated to EQ, NQ, LS, LQ, GR, GQ)  
.....

SUM  
COMPUTE AVERAGE item WHERE etc.  
MINIMUM  
MAXIMUM  
.....

SHOW item. This means that the list of unique values associated with the item will be shown.  
.....

Under development, but not yet operational, are LUCID system programs operating under Time-Sharing that will update, merge and subset LUCID data bases.

In the planning stage, but not yet designed, are a report generator, other query languages, and experimental redesign of internal data structures.

A variety of data bases have been produced with LUCID, GORed and then QUIZed. The general categories of information handled have been: accounting, inventory, census, phone book, personnel, and damage assessment.

Documentation:

TM-2354/001/00	LUCID USERS MANUAL (SDC document)
TM-2429/000/00	QUIZ USERS MANUAL (SDC document)
TM-2302/000/00	GOR USERS MANUAL (SDC document)

GENERAL PURPOSE DISPLAY SYSTEM (GPDS)\*

Sally Bowman  
GPDS Project Leader  
System Development Corporation

The General Purpose Display System (GPDS) is an experimental, computer-based system for the generation and manipulation of cathode-ray-tube (CRT) displays. GPDS is designed to permit nonprogrammer users to retrieve numeric or textual information from a data base and to display the data in almost any format on a CRT. The GPDS user interacts with the program system by replying to English language queries presented to him on the CRT. As he replies to these queries, the user constructs a special-purpose program called a process, which he may then recall to bring about the retrieval, display, and manipulation of his own file of data.

The General Purpose Display System consists of hardware, a language for communication with the program system, and the program system itself, all operating under the control of the SDC Time-Sharing System on the AN/FSQ-32 computer.

For minimal operation, GPDS requires a teletype and a CRT. The teletype is used for communicating data, for responding to some of the program system's queries, and in the construction of the man-machine dialogue. The CRT displays data. A light pen attached to the CRT is used to reply to multiple-choice queries and to make selections from the various control lists presented to the user. A button box, containing some 60 buttons, calls common control functions or generates control displays facilitating the dialogue between user and program. The standard GPDS console also includes a RAND Graphic Input Device (GID), which permits designation of points on the CRT with a resolution of about 100 lines per inch. Additionally, an auxiliary keyboard is included, permitting use of the standard teletype for communication with Time-Sharing and the auxiliary keyboard for communication with GPDS.

The second element of the General Purpose Display System is DUCOL, Display Users Communication Language. It fulfills three specific needs. First, it provides the user with a facile way of constructing man-machine dialogue that is meaningful within the user's context. The user is able to construct English-language queries as well as enter commentary describing his process. Secondly, DUCOL provides a powerful and highly descriptive retrieval language enabling the user to obtain facts required from the data base, in terms that he can readily understand. He might, for instance, retrieve the names of all cities from a census data base in the state of Illinois where the population for the year 1950 lay between 50,000 and one million with a single statement. DUCOL's third facility is its arithmetic capability, which has all the power of ALGOL.

---

\*The development of this system is supported by the Advanced Research Projects Agency, under contract SD-97.



Data can be retrieved according to mathematical criteria and displays can be manipulated arithmetically. Additionally, conversions are performed automatically among any of a wide variety of units prestored in the system. The user may perform complex operations in varying terms (i.e., inch, foot, meter, mph, degrees, dollars), and the program system will perform conversions automatically, informing the user only when his result is noncomputable (for example, inches into degrees).

The third element, the initial program system, consists of an extensive control complex, a DUCOL interpreter, and approximately 20 small, independent operating routines called primitive processes. Each of the primitives accomplishes a simple function--draws a line or a circle, displays textual information, rotates or scales a display, or facilitates iteration control. The primitives consist of a set of queries for communicating with the user and operating code to perform their functions. The dialogue between user and system begins as soon as he requests the operation of the system. In response to queries from the system, he furnishes information about equipment, constants, and the data base he will use. He then selects with the light pen the name of an existing process from a list of those displayed on the CRT. The dialogue continues as the user builds his new process. The user may replace the language of any query with language more appropriate to the process he is building. For example, LINE's query "Supply starting point for line" might be replaced by a user building a bar chart with "Supply origin of bar chart." The user may also make the reply to any query permanent, so that when he operates the process now being built the query will be suppressed and the answer supplied automatically. As each requested operation is completed, the user specifies whether to save the new process as it is, review the new process, forget and start over, or add another process. In this fashion, by replying to queries presented, by altering queries to language more meaningful in the context of the process he is building, and by storing permanent answers where appropriate, the user builds his own processes from one or more of those available as initial building material. When he has completed his new process, he gives it a name meaningful to him, supplies descriptive commentary, and saves it. Now when he calls for the list of available processes, his new process will also appear and he may use it to accomplish its specific purpose or as a building block for another new process.

Commonly, data retrieved and displayed by GPDS would come from a large and potentially powerful data base previously made up by LUCID and organized by GOR. (see p. 4-3). Alternatively, the user may enter his data on line, through the teletype or the auxiliary keyboard, and store the data permanently for subsequent display and manipulation.

Although the full range of the communication language DUCOL is not yet implemented, most of it is available and has been successfully demonstrated. The retrieval capability has been used for demonstration purposes. The primitives that draw lines and circles, reproduce data from the RAND GID, rotate and move figures and textual information on the CRT, manipulate textual information, sort and sum data, provide for core storage of selected data and perform iteration control are in daily use. All processes built to date have been for checkout or demonstration purposes, but the mechanisms in the control package for creating, saving, altering, and recalling processes are functioning successfully.

ECCO and EPIC

Roderick H. Bare  
Manager, Personnel Data Center  
System Development Corporation

I. INTRODUCTION

A three-phase program to develop, implement, and test a time-shared personnel information system for System Development Corporation was initiated in March, 1964.

The first phase involved developing a prototype personnel data retrieval capability to operate within the ARPA-sponsored time-sharing system. The retrieval program system, called ECCO, was released for operational use in June of 1964. It is currently being used operationally by SDC and the Job Corps, and experimentally by the Office of the Assistant for Personnel Systems for the Air Force.

Phase two of the project was concerned with (a) testing the utility of the prototype system for meeting operational requirements, (b) obtaining inclusive statements of requirements for personnel information from users of and customers for personnel information, and (c) designing, developing, and implementing the Experimental Personnel Information Capability (EPIC).

Phase three of the study is now beginning. It concerns the study, in an operational setting, of the functional, behavioral, and organizational implications and effects of a time-shared personnel information and management information system.

Both ECCO and EPIC operate on the SDC time-sharing system using the IBM AN/FSQ-32 computer and teletypes. Each program provides an on-line inquiry capability for searching personnel data files and outputting the required information. Within the limits of the program, the teletypes allow the inquirer to insert the commands, control information, and search parameters required for the data retrieval. It communicates the results of the search to the inquirer and allows him to specify the format of the desired output and the arithmetic operations to be performed upon the data.

II. ECCO CAPABILITIES, OPERATING DESCRIPTION, AND DATA BASE STRUCTURE

The ECCO program system provides the user with a retrieval system for selecting on items or combinations of items from the data base. The data base is not limited to any specific content but rather is identified by the first record, which indicates what the items are and where they are located in all succeeding records. The ECCO program is written in the JOVIAL Time-Sharing (JTS) language, and requires approximately 17K words of core storage for operation.

### A. Capabilities.

1. ECCO has a generalized search capability whereby individuals with various backgrounds and characteristics may be identified from the data base. Individuals are selected on the basis of search criteria, and all records meeting these criteria are stored in a special disc file. The full records for those persons who qualify (i.e., match on the specified criterion variables) are temporarily stored either for subsequent searches on this subgroup or for a listing of information on these persons. It is also possible to locate all information about a single employee simply by using the employee's man number as the criterion variable.

2. Once the search has been completed, a list capability is provided which allows the operator to specify a printout of any of the items of information contained in each individual's record. It is possible to call for any item of information in the individual's record, or any combination of those items, once he has qualified on the search criteria, even though these items of information were not included in the search criteria. The user may vary the items to be listed. The format lists the employees selected in the order in which they appear in the data base.

3. A capability is provided to perform four statistical operations on any quantified items of information in the data base. The statistical routines are the arithmetic mean, range, standard deviation, and sums of squares.

### B. Operating Description and Data Base Structure.

The first record of the data base describes what the remainder of the data base consists of, the number of characters per item, their location, etc. The remaining records in the data base consist of the specific items of information on which retrieval is desired.

Individual records must be of fixed length and fixed format. The number of characters in an individual record must be a multiple of 8 and should not exceed 600. Individual records should be recorded in blocks or physical records as large as possible for efficient processing. The maximum size is 32,760 characters per block. An individual record may be just one physical record but may not be split between two physical records. Alphabetic titles for items are included in the data base for output but not for selection purposes.

The label record is in three parts: a label card, a deck of dictionary cards, and a deck of list format cards. All must be in the same physical record, not as one card per record. The label record may not be more than 32,760 characters in length. There may be a maximum of 99 named data base items or fields in an individual record.

### III. EPIC CAPABILITIES, OPERATING DESCRIPTION, AND DATA BASE STRUCTURE

EPIC provides the user with a more powerful capability for retrieving, manipulating, analyzing and reporting personnel information than ECCO. Through an on-line programming feature, the user is able to manipulate the data base almost

without limit. However, the on-line programming feature temporarily has made the system less customer-oriented and more programmer-oriented than the ECCO program. Eventually, by storing the most frequently used on-line programs in a library, the user will be able to call up many of the procedures automatically rather than having to program them each time; thus EPIC will evolve more and more in the direction of a customer-oriented system. The EPIC program is written in the JOVIAL Time-Sharing (JTS) language and presently requires approximately 38K words of core storage for operation.

#### A. Capabilities.

##### 1. Record selection, elimination, alteration, contraction, expansion.

These features permit the preparation on-line of a special data base containing just those items of information and those individuals in which the investigator is interested, thereby eliminating the need for processing the master file, and reducing the amount of time required for processing. These subfiles may be stored on disc or tape for subsequent processing.

2. Time-slice function. This allows the user to set the data base so that it exactly replicates its structure for any time in the past.

3. On-line programming. This provides the ability to process the data in a wide variety of ways including statistical computations, comparison of values, development of derived items, sorting, and variable output formatting.

4. Record flagging. This permits the user to flag one or more records about which there is some special information and thereby produce an output indicating what that special information is or where it is located. The flagging procedure may also be used to identify items or records in the data base on which a special operation is or is not to be performed, without actually having to subset the data base.

5. Correspondence tables. The output of alphabetic information such as titles, names, etc., is done by means of these tables. By removing the titles from the data base itself, the records are reduced in size, with a commensurate reduction in the processing time.

6. Simulation. The user can insert values in the data base with future effective dates and then test these values as if they were current or historical items of information.

#### B. Operating Description and Data Base Structure.

EPIC derives its general-purpose capability through the use of a library record which is quite similar to the label record used in ECCO. The program can operate on the items in the data base despite the fact that the data base consists of unequal-length records. The content of the data base is irrelevant to the operation of the program, and EPIC is therefore not limited to operating on personnel information.

SECTION VREPORTS OF THE WORK GROUP CHAIRMEN

Seven concurrent work sessions were held to enable the participants to discuss, in some detail, a number of important problems in the data base area. On the first day, the groups met for an hour to get acquainted and to have the chairmen lay out the problem. They then met for three hours on the second day to probe the problem in some depth and, if possible, suggest solutions. Judging from the reports of the chairmen, the discussions were stimulating and productive.

Criteria for Going On-line.....	5-3
Entry and Query Language Design.....	5-7
File Organization.....	5-9
File Sharing and Protection.....	5-13
Theory of Data Base Problem Definition.....	5-19
Criteria for Evaluating Data Management Systems.....	5-23
Recording for Analysis, Costing, and Control.....	5-27

## CRITERIA FOR GOING ON-LINE

Chairman: Ralph G. Tuttle, Office of Naval Research  
SDC Associate Chairman: Robert A. Mosier

Although the use of on-line systems is not new, the tremendous potential advantages promised by the concept of on-line operation, in its application to a much wider group of data base systems, is currently attracting increasing attention. The existing need for the promised capability is so great that feasible solutions to problems arising from these applications are important and urgently needed. The work group attacked the problem of identifying and describing those system requirements which can be met sufficiently better by on-line capability so that the selection of an on-line mode of system operation is justified.

The work group viewed the problem from the eyes of a project manager having the responsibility for providing system capability, the authority to select the type of system to provide that capability, and the resources to pay for it. The purpose of generating the subject criteria is to provide such a manager with basic information on which he can make a judicious selection. This information should be in the context of:

- Usefulness (does he need it?)
- Technical Feasibility (can it do his job?)
- Cost (does its effectiveness justify its costs?)

The time available in the working session was much too short to identify and describe criteria adequately. Nevertheless, the ideas expressed and considered proved to be beneficial to the participants and may be informative to others interested in establishing criteria in a more deliberate manner.

In establishing ground rules for the discussions, it was agreed that the related subjects of time-sharing, real time, and multiprogramming would be discussed only as they apply directly to establishing criteria for selecting an on-line capability.

One initial thought expressed is not followed rigidly in the considerations presented below, but does deserve attention in future attempts to describe criteria. Actually, future batch-processing systems may well have at least one interaction console, and several input-output channels and buffers to which additional consoles may be connected. We would then have batch-processing systems which have at least some on-line capability. We also know that batch processing can be performed by on-line systems. The question, then, should not be: "Should we select an on-line system instead of a batch-processing system?" but rather

"Which functions of a hybrid system should be performed on-line?" The selection criteria, then, should be addressed to the extent of on-line capability that can be built into a system justifiably.

It was quickly agreed that on-line requirements will differ significantly among systems supporting different missions assigned to different types of organizations solving different problems in different environments. The development of adequate criteria must identify and consider the commonalities and differences among the requirements of these various systems.

Response time and penalties for delay appeared to be important criteria. The criteria may differ in extent when applied over the wide spectrum of degrees of concurrent interaction required among the total group of system users in possible on-line applications. At one end of this spectrum each user is using a mutually accessible data base, but is working on individual unrelated problems. At the other end, all (or a significant portion) of the users are working on different but related elements of a common problem having a single deadline for its solution. In the latter case, the elements of the problem require the solution of intermediate steps of other elements as inputs, and may require iteration procedures among element solutions.

For individual users working on unrelated problems, the speed of response should be commensurate with the human's physical and mental capabilities. The penalties for not providing this speed include: delays in completing the individual tasks; annoyance suffered by the user which could encourage him to adopt other means of solution; and reduction in benefits to be obtained from the symbiotic relationship between man and machine which depends upon spontaneity of response. The extent of these penalties is associated with factors such as total number of users, system size, frequency, and length of time per use per user. These factors themselves constitute important categories in which criteria can be established.

At the other end of the spectrum of user interaction, individual users conduct their problem solution concurrently and in parallel. Outputs generated during the solution of one problem element by a single user are required continuously as inputs to other problem elements being solved by other users. For example, the strongly related operations, logistics, intelligence, and communications annexes to a military operational plan are prepared in parallel under a tight time schedule by users having access to a common data base. In addition to the criteria developed for the individual user, additional criteria are necessary in this case to reflect the critical response speed required to permit efficient data management, so that the effectiveness of the entire user complex is not seriously impaired.

A decision as to whether or not to go on-line for a system having remotely located elements is critically affected by the capability of the connecting

communication system. Procedures, hardware, and software required to provide rapid, secure, reliable, and available computer-to-computer interaction pose important problems for such a system.

Cost is always an important criterion and is treated as a topic by another work group. A few points introduced in the criteria discussions are presented here. Total system costs must be used as criteria, and these can be measured in dollars, time, and manpower. The training and availability of operational and maintenance personnel are important considerations. In general, the training problem does not appear more serious for on-line systems than for batch-processing systems. On-line systems require more operational availability than batch-processing systems. Consequently, additional component redundancy is required for on-line systems and may increase costs. On the other hand, the use of on-line procedures may decrease the requirements for the conventional types of input-output equipment and thereby reduce costs. The large advantages of on-line computer programming and debugging are just beginning to be realized. Significant cost reductions in these tasks are being realized by performing them on line. A 3-to-1 cost reduction in program debugging is reported by some studies.

Although technology has long been available for on-line systems, the increased attention being given to the systems may produce significant new technology, especially in the software area. In determining when to go on-line, system obsolescence, growth potential, and constraints are important. The areas of security and propriety protection require additional research and development for some types of systems.

The topic of "Criteria for Going On-Line" is large and complex. It is considered to have sufficient importance and urgency that the 3rd Symposium on Computer-Centered Data Base Systems could use it as the central topic. This is strongly recommended.

---



## ENTRY AND QUERY LANGUAGE DESIGN

Chairman: Professor Frederick B. Thompson, California Institute of Technology  
SDC Associate Chairman: Levi J. Carey

The work group consisted of representatives of government and industry, managers and contributors, users and developers, experienced personnel and those new to the field. Therefore, it is surprising and significant that a reasonable degree of agreement was reached regarding the subject under discussion.

A sharp distinction was made between the structures in the data relevant to the user, and the organization and manipulation of the data in the computer. The languages used to communicate with the computer should not be limited to terms of the file structures, data sets, and memory organizations within the computer. They should permit the flexible expression of structures in terms that the user would naturally and relevantly look for in the data. The terms used by the language should arise from the subject matter of the data rather than from the formal representation in the computer.

Although the language should be relevant to the user, this does not imply a highly sophisticated language. No requirement is made that the language be like English. Indeed, discussion of natural language was noticeably missing in the work group. This is not because a more highly formatted and stylized language is satisfactory, but rather because more rudimentary and direct forms of communication are more appropriate. In particular, direct communication with the computer at a low level of syntactic complexity is more to the point than complicated queries. Querying should be reciprocal in centering down in the ultimate answer. Light pen and button can be direct and effective instruments of communication.

This work group considered that the ability to get at the data base in unanticipated ways, in ways that might well cut across any preconceived organization of the data base, is where the immediate advantage lies. The query expresses an anticipated structure; the projected structure will not materialize until the answer to the query is elicited from the data base. Therefore, it is in the capacity of the query language, or query procedure, to express structural relationships relevant to the users' mode of thought that a data system finds its applicability; and it is in the capability of the system to respond to such queries that it finds its power.

Expanding on this nucleus of our consensus, we considered the following question

What about efficient utilization of the system and computation time?

The ability of a user to approach his data in novel and unanticipated ways, and to elicit from his data the aspects and relationships relevant to his transient but pressing concerns, has such great value that it

outweighs whatever incremental cost increase might accrue from this mode of operation. However, increased costs are not anticipated. Batch-processing and procedure-oriented systems incur their own inefficiencies; in particular, the loss of time and relevance to the user. The increased ability to narrow down on the relevant computations is expected to compensate for the inefficient internal organizations necessary for more flexible processing and freer modes of communication.

What about the inclusion of complex processing capability, such as powerful algorithms?

Those who are unable to use advanced analytic techniques cannot profitably use algorithms. Those who can use advanced analytic techniques have developed specialized techniques tailored to the requirements of their subject. Algorithms useful to them on the computer are usually available. The problem, then, is to elicit from the data base the material which they now need processed by these algorithms. To a system which will elicit these latent data, one can add the simple task of applying existing analysis programs. These programs need not be built into a complex language system to be useful relative to current practice.

What about growth, extension, and specialization of language in response to the user developing an understanding of his problem?

This question is typical of concerns that occur to those developing advanced systems. The multiplicity of directions for improving the communication capability with the machine requires some direction from the user. However, at present, we do not have adequate experience with the type of system that allows free and subject-oriented access into the data base to give any such guidance.

The increase in usefulness of these systems appears to us now to be so great and our experience with this new capability so limited, that the group could not determine what the next step might usefully be.

The title of the work group includes reference to entry languages. However, at the present time, our problems revolve around large streams of data that flow into the system and dictate their own entry formats. Thus "entry language" is not a pertinent consideration. Finding and getting at relevant structures in these bodies of data is more important than trying a variety of structures on the data body as a whole. The group agreed that there was a need to be able to "massage the data," but in the sense of reaching into an apparently amorphous mass and seeking relevant threads of structure, in contrast to taking the data in one's hands and molding it into possibly relevant shapes.

Above all, we all felt the lack of experience with such data systems. We need more such systems and more experience with the ones we have. On-line capability to deal with the data in terms of the natural immediate aspects that arise from its significance to the user has the big, immediate payoff. There were a few systems of this kind represented in the group, and the experience with these systems is verifying the conclusions stated in this report.

## FILE ORGANIZATION

Chairman: Jack F. Nolan, Lincoln Laboratory, M.I.T.  
SDC Associate Chairman: Tom J. Koeske

The discussion for the work group centered on the basic idea of providing special information within a large data base system to describe the internal structure of the data files. The purpose of internal file descriptions, of course, is to enable the system to be more responsive to changes in the data structure and file-processing operations.

The way in which files are structured has important effects upon all functions of the system (much in the same way that item packaging in the production industries affects distribution, sales, inventory, and eventually profits). There is a danger, in data management systems, of basing the system design upon rigid preconceptions of the contents of data files and the required file-processing operations. The difficulties that arise when these preconceptions are wrong (or are only temporarily valid) have been well documented. There have been many approaches to introducing versatility in system operations to adapt to such changes; they all, in some fashion, must begin by incorporating file structure information within the system.

Basically, this work group was concerned with the following problem: "What sort of information does the system have to have and in what form to allow users of programs to make symbolic access to data in new and unpredictable ways?" This symbolic access should be available, not just on a file level, but on the level of actual data items (the data fields).

The work group reviewed briefly the approach in internal file description in four current systems: SDC's LUCID, MITRE's ESP system, Informatics' File Management System, and an experimental Lincoln Laboratory system. Three of these systems, that is, all but the Informatics system, are essentially on-line systems and operate on a facility that is time-shared between many concurrent jobs. The distinction between these two modes of operation (batch processing of large sequential files, and time-shared access to parallel files) indicated a requirement for markedly different file structuring.

The discussion indicated that a limited amount of cross association of data is required in the batch-processing systems. In serial-processing batch systems there is an interest in hierarchies of data, but this arrangement is largely a local property (within records), usually built into the files from the beginning and not changed dynamically during processing. On the other hand, the time-shared systems require a high degree of cross association of data within files and means for unpredictable new associations of data to be set up by the users dynamically as they operate. The batch systems are also, of course, characterized by very high volume outputs (report generation) while time-shared systems

are more concerned with a large number of low volume accesses to data files (fact retrieval). Batch systems operate on data that is fugitive to the system (normally stored off line); in time-shared systems the data is resident on line and is, therefore, always under the control of the system. For performance criteria, batch processing puts major emphasis upon efficient machine utilization; time-sharing places the emphasis on response times.

In batch-processing systems one is frequently concerned with handling files of external origin; thus there is usually an interest in minimal constraints upon the class of files that can be accepted and processed serially by the system. On-line time-shared systems, as in military applications, have selected file structures as part of the system design for real-time service to a particular class of users in a particular application. The general-purpose, time-sharing system in which data filing and access needs are essentially unpredictable represents a different case. In this instance the problem has been "solved" by these systems by having the files stored by file names only, and by making no attempt to provide the user with a detailed control upon the internal contents of the files. (For this reason, the full power of time-sharing has been limited to users who are trained programmers.)

All of these differences are, of course, reflections of the different character of the applications with which they deal. The discussions indicate that because of these differences, present limitations upon the time-shared system and the batch-processing system are quite different. What, for example, would be the gain in batch-processing systems of additional sophistication in internal descriptions of file structures? The answer appears to be "relatively little." In the systems of this type that were discussed, a fairly general flexible language has been developed for describing the serial files, for automating the process of defining a computer run, and for outputting the data into prescribed formats. Although not in general practice, the required techniques have been demonstrated. Batch processing of serial data is primarily limited by serial input-output data rates, and this limitation is basically a hardware problem.

An improvement in performance could be expected by improvements in multi-programming techniques rather than in the software to support the file operations. If a batch-type operation is converted to operate as part of a multiprogrammed system, it is probable that a fresh approach to the organization of the files could be made, even in routine business processing. Serially organized files are largely the result of the serial nature of the processing systems. For example, when one begins to operate upon data on-line in a multiprogrammed, parallel-access environment, all data qualifiers may have equal potential as search criteria.

In contrast, designers of time-shared systems feel that present data management techniques fall far short of user requirements. In particular, the techniques are weak in allowing dynamic association of data within and across files according to the user's special needs. Hardware that has been specifically designed

to allow the efficient multiprogramming needed for time-shared operations is now becoming available. Data management techniques to exploit these hardware changes were briefly considered during discussions. For example, the introduction of memory protection, highly parallel input-output systems, dynamic relocation, and paging techniques (and the associated concept of the use of virtual memories) are bound to have a major effect on data management techniques. The design of the file structure to operate in such systems has to be intimately related to the way in which the hardware will perform these functions. To fully exploit these features, the system must take on more responsibility from the user or user-program. For example, the user will have little understanding of the actual locations of his data by volume or even by the class of auxiliary storage. One can expect to see a great deal of overlapping of multiple input-output operations with less system penalty for sequential access as long as there are enough jobs active on the machine at one time to keep the facilities operating in parallel. One can expect an increased use of list structuring of data in order to effect the association of data. Because of the special nature of the interaction of data management with paging and relocation hardware, list structuring will be essentially different in character than that which has been developed to operate entirely within a random-access memory. Techniques must be developed to provide for efficient dynamic cross association of data within a hierarchy of memories.

The work group also discussed the question of whether there is anything basically incompatible between the problem of large-volume batch operations and on-line time-shared operations. Apparently time-shared systems that exist today do not do batch processing very well. The consensus of the group was that this lack is not fundamental to the design of the systems, but results from a relative lack of attention to the problem of their combination. To date, time-sharing effectiveness has been demonstrated only within the context of use by experienced programmers and within an environment not critically pressed with real-time requirements. One of the developments of interest for the immediate future is the effective combination of large-volume processing and those of conversational time-sharing.

## FILE SHARING AND PROTECTION

Chairman: Richard G. Canning, Canning Publications, Inc.

SDC Associate Chairman: Lee S. Christie

This work group decided that the subject should be limited to a consideration of systems that serve many on-line users. In such systems, the questions are: (1) "What are the problems of user sharing of files of common programs and files of community data?" and, (2) "What are the problems of protecting the files of multiple users from both inadvertent change and deliberate but unauthorized access?"

The work group recognized that, up to the present time, almost all on-line, shared, interactive usage of computers has been for individual work that is done in parallel. Coordination among users working on a joint task has been on a man-to-man basis. However, interactive computer-assisted task integration by organized groups pursuing a common problem may become common in the near future. The use of the computer to support the task-oriented integration of a group was felt to be an unexplored area for research and development. An urgent need exists to investigate this area; file-sharing investigations offer one good point of entry. The group spent no effort in this area because the state-of-the-art is at such an embryonic level of development.

On the other hand, file privacy has been investigated for some time. It has been pursued in time-shared systems by making each file anonymous to all except its owners. This system is satisfactory if other system users (authorized or unauthorized) are indifferent to the file, and inadvertent disturbing of the files can be prevented. It is not sufficient if security rather than mere privacy is at stake, since indifference in that case is ruled out by definition.

To understand the approach used by the work group, the subject may be viewed as being arrayed in two columns and two rows (see figure below); the columns are headed "Read Only Retrieval," and "Retrieval for Update;" the rows are designated "Content Validity" and "Content Security."

	Read Only Retrieval	Retrieval for Update
Content Validity	Case A	Case B
Content Security	Case C	Case D

These last two items deserve further definition. Content security is concerned with who reads the file and who changes it, regardless of the file contents so long as their protection is required. The concept involves the ideas of commercial or military security. Content validity is concerned with what is read or changed, regardless of who is doing it so long as the contents matter to the user. This concept involves what epoch or generation of data it is that is being retrieved. Does it all represent the external state of affairs at one point in time? What was the time of the last updating of particular records?

The various combinations of these headings in each of the four cells shown in the figure (Case A through Case D) enable the four possible combinations of factors to be discussed individually. The first cell of our array, Case A in the diagram, is the "Data Validity, Read Only" situation. In this the group could foresee no problem; no special control or protection techniques appeared to be involved. Such a file is totally replaced if it becomes invalid, rather than being internally changed. So once (i.e., initially) the validity is adequately assured and back-up is provided to guard against catastrophic failure, no further concern for the on-line situation need be expended. Program files that are in frequent and repetitive use are often of this character.

Case B, "Data Validity and Retrieval for Updating" presents a problem of control of updating of community data where several people might be wanting to update the same data "simultaneously." In this situation two different conditions were identified. One condition is represented by the systems used to make airline reservations. The work group considered that the solution to this example was well worked out and operational. Although many remote terminals update the same files at the same time, in general these systems apply to a single file of data (inventory records of seats), the transactions are handled rapidly, and, because a transaction ties up records one-by-one and for a very short period of time, updating causes no interference.

The work group identified a broader problem: that in which multiple, simultaneous users of the same data might use various records for more lengthy periods of time. Some of the characteristics of this condition are as follows:

The user might be at the console working with the contents of a given file for a long period of time (compared with the expected interarrival interval between inquiries for the same records). Updating an entry in the data base might involve several interrelated files rather than just a single one, and the user might not be able to predict this ramification when he sits down at the console. He might not even be able to predict whether or not he will update the data base, possibly needing to observe the contents in order to make this decision. The problem arises when two or more people attempt to use the same files under these conditions. A situation of this type could occur if several people were working on the same project and attempting to meet the same deadline.

Some possible solutions to this validity control problem were discussed. One solution would be to process transactions in a batch mode and to maintain a change file that is queried before the main file is queried. This arrangement would exert a discipline on the system that could effectively eliminate mutually invalidating changes to the same data. Care would need to be exercised that the contents of the change file, when processed against the main file, would make changes to the record in such order that the resulting state of the main file is truly up-to-date. The best order would not necessarily be by time of receipt of the change. For example, banks and customers both prefer that when a deposit and a check that could create an overdraft arrive simultaneously (same day), the deposit has priority for processing. This problem of order of processing can be difficult in the case of multiple, ramifying effects of a single change on interrelated files, but, in principle, it can be done. However, this solution can get out of hand if changes are arriving at a high rate relative to the update interval. The change file could need its change file and so ad infinitum. Moreover, this solution would be expensive in terms of both storage space and response time, and might be unacceptable on these grounds alone in an on-line system.

Another solution might be to lock out any further queries during a retrieval interval when such updating could take place. In this case when a record is retrieved, no one else could make changes to it until the first retriever has released it. This solution also might be unacceptable in many systems because the response time could be affected appreciably. Further, it seems unreasonable to deny access to data that is about to be changed since the current data may be quite as satisfactory to its user as the new data if the alternative is a "busy signal."

A third solution suggested was that records be identified as to the time they were last updated. Then when a record is first retrieved, the time of last updating would be printed (or displayed) right along with the record, and this tag would then be used, in effect, as a lockword for controlling the permission to modify the record subsequently. Access for read-only would not be affected, only the entry of a change. For example, if someone were to enter an updating change for a record, he would first enter what he believed (from his display) to be the time of last updating and the computer would check to see if this agreed with the current value of that field. If so, the change would be processed; if not, the computer would notify him that the record had been changed in the interim. He would reconsider his change, modify it if necessary, and enter it using the new time "lockword." There would be a modest storage cost to this proposal but no deleterious effect on responsiveness.

The work group was aware that there were probably other and perhaps better solutions, but time did not allow further investigation. However, the third solution appeared so promising that the SDC participants in the work group proposed such a change for the SDC Time-Sharing System.



Case C, second row of the array, involves Content Security and Read Only Retrieval. This is the "read-protect" function involving the problem of privacy. The work group discussed a number of solutions to this; the state-of-the-art seems to be fairly well advanced. One solution is to use lockwords for the hardware, software, or both. The software lockword would be stored with the record. The user would have to submit the lockword with his request and the lockword would have to match up before the user would be allowed to retrieve the data. The lockword would either be fixed, that is, a set value for some lengthy period of time, or variable. A person's name might be used as a lockword although there was discussion as to whether this made real sense since a person's name (to serve its normal functions) must be public knowledge.

The use of authority lists and requirements to have the computer identify what a terminal is authorized to receive information was discussed as well as the question as to when identification of the terminal is adequate and when it is necessary to identify the person who is at the terminal. The work group decided that solutions to either of these cases could be handled by hardware, software, or both. An example, of this type of safeguard would be to have the computer break the connection initiated from a console when a request for data is received and make a new connection initiated by the computer before data is transmitted. Another arrangement that was discussed briefly was the use of supervisory keys for consoles; that is, certain transactions or inquiries would be honored only if the correct supervisory key were inserted in the console.

One means for controlling violations of access security that was suggested was to have the computer count the number of unsuccessful retrieval requests within a period of time to find out who (or, at least, what terminal) was making an attempt to get a piece of data. If this type of solution were used, it was pointed out that a lockword system with fewer combinations could be employed. Not as many combinations in the lockword would be needed if checking on the number of retrieval requests that are made for a given record were constantly going on. Another and more powerful facet of this solution would be to limit the number of retrieval attempts within a period of time--three attempts and the connection would be broken, and, in case three attempts did occur, someone in authority would be notified.

Message scrambling was also suggested as a safeguard. Programmers working on checkout could run data of proper form for checkout through the program, but the content of the data would be scrambled before it was printed, punched, or in any way put in a form readable by the programmer. One of the members of the group reported that this has been used successfully at her installation and was more satisfactory than using dummy, or simulated data.

The following shortcomings of these solutions were discussed:

The protection systems of which the group was aware typically use only a single condition to be met in order for data to be retrieved. All one has to know is the lockword. Protection systems in the future may need multiple-condition tests, such as having two lockwords based on different encrypting schemes, or a lockword plus an authority list, or some other combination of things. Another shortcoming is that program changes might inadvertently invalidate some of the security measures, thereby opening up loopholes. Communication lines themselves might offer loopholes for compromising the data. Communication lines might be tapped, a store-and-forward message center might forward the data to the wrong terminal, and so forth. These areas are weak points in the system.

Case D in the array, that of "Data Security and Retrieval for Update," is the most complex case encountered. The problems involve both security protection and correct updating (Cases B and C) so the group felt that Case D was covered simply by conjoining the considerations involved in the other two cases already discussed.

Other points that were discussed briefly had to do with the solutions that are available for data backup or data in mass storage units. The following solution was described:

Two copies of the file are created originally, one that stays on-line and one that is stored in a vault. This method assumes that the file is maintained on a replaceable cartridge, disc pack, or similar medium for storing the data. As the on-line file is updated, a copy of the updated record, together with the mass storage address, is stored on magnetic tape. At the end of the day, the second copy is removed from the vault and put on the computer. The tape is played against this second copy. The second copy is updated by a simple replacement process. One then has an updated second copy of the file which can be put back into the vault.

Conventional security practices are satisfactory for off-line vault storage of removable files. The need to monitor the protection system was discussed. It is possible that indiscriminate use might be made of the protective system to hide data, and it was recognized that indiscriminate use of protection can seriously impair the usefulness of the data base. The work group also briefly discussed the need for a by-pass system to get around the protection. A by-pass system is a risk, of course, but apparently the risk must be taken. For instance, if a lockword is lost, some means will be needed to retrieve the record in the absence of the lockword. The work group also discussed the need for changing the lockwords at specified intervals and also upon termination of an employee who knows some of the lockwords. No special problem was identified in distributing new lockwords to all necessary people; the distribution would have to be done in accordance with standard security procedures.

Another subject discussed was the question of clearance and fidelity bonds for people who might have access to sensitive data (programmers, operators, customer engineers, etc). This solution to protection problem is one that has not, to

our knowledge, been used widely in the commercial field. Military security of data might include all of the above requirements, where they would probably be applicable to a lesser degree in commercial applications. In addition, military security will usually entail some additional requirements, such as verifying the requester's need-to-know before the data is retrieved. To satisfy military security requirements, protection measures implemented internally in the combined hardware/software system must be such that, within the total system (including the users), the weakest security link is the users and not the hardware/software part of the total system. In addition, the character of the user weaknesses should not be changed by interaction with the hardware/software system so that known personnel security measures are reduced in effectiveness.

The group knew of no significant research being done in file-sharing and protection. References that exist in literature tend to be superficial (for example, dismissing the problem by saying simply that lockwords are the solution). Also, although airline reservation systems have been in existence for more than five years, little is known about how problems represented by Cases B, C, and D have been solved. The group agreed that additional discussions on this subject would be most desirable and, in fact, that a survey report on present accomplishments, should be published in some such medium as the Communications of the ACM.

## THEORY OF DATA BASE PROBLEM DEFINITION

Chairman: John W. Young, Jr., National Cash Register Company  
SDC Associate Chairman: Stanley L. Kameny

The title of this section is subject to two interpretations. The first view considers that the title refers to a theoretical base for the process of analyzing a problem, i.e., for the steps through which one goes in working out the problem description. The second takes the title to mean a theoretical framework in terms of which an information processing problem can be described in a static fashion, i.e., a framework that is related to recording the results of an analysis of a problem. These views are not incompatible--processes used to analyze a problem can be related to the various aspects of a theory of problem description.

Let us consider the steps in an idealized communication between a user and the information processing system (including the human analyst, if any) as they work together to describe a problem. This process is not strictly linear, of course, but is actually highly iterative.

The analyst (human or machine) begins by establishing the set of data elements involved in the user's problem. These elements are all the entities and attributes (i.e., characteristics) with which he deals. Some of the detailed considerations in this phase will be discussed later as part of our tentative start toward a theory. Secondly, the relationships among the data elements have to be determined; that is, the fixed correspondences (such as that between Employee Number and Employee Name) which exist independently of any processing have to be noted. In many cases, of course, the information about these correspondences will be obtained at the same time that the elements themselves are being defined. These two steps now have defined the structure of the data base.

The third step in this process of problem analysis is to determine the user's requirements for processing--what are the external events whose occurrence is to be reflected in the system, and what is the information which must be provided by the system, either on a schedule or on demand.

The fourth step is to trace the user's requirements back through the data element structure to see how his required information can be derived, and to locate additional data elements which may have been left out of the original set.

The fifth and final step is to establish the processes by which the entire system can conveniently be modified as the user's problems and his understanding of them change. All of this communication between the user and the information processing system must be in some language, possibly natural English. We can hope that a theory of problem definition, although probably not directly applicable as a language (e.g., Information Algebra), would aid in the design of the basic structure of a suitable means of man-to-man-or-machine communication.

Such a theory might also aid in the development of systems which tell the user that what he is saying is ambiguous, inconsistent, or not consonant with a pre-defined set of data elements; the systems might then ask him to clarify his statement or might indicate how the existing structure must be changed to meet his problem. Unfortunately, precision in a natural language apparently is gained only through verbosity.

The work group emphasized that great care must be taken to shape the system, certainly in practice and perhaps in theory, to the user's viewpoint, requirements, and language, rather than to ask him to conform to some sort of original structure which pleases us. In fact, there seems to be a strong tendency to develop meta-languages for language description, rather than languages themselves, so that the user can communicate in any way he likes, and the system will accommodate him.

A characteristic which the work group members felt was most important and which was lacking in many present systems was the capability for dealing with problems which at a given moment are incompletely or inconsistently defined. A problem description technique (theoretical or practical) should be able to cope with situations in which operational pressures require that a system be implemented before the problem analysis is complete, and certainly before it can be certified to be accurate. We need a way of describing a problem "approximately." We also need a capability in data base systems to handle the situations that occur when a discrepancy is detected between the problem as described and the usage as encountered--we can envision systems which say to the user, "The information you have asked for is not in the system under that name; could you possibly mean...?" In addition, we need systems which will reorganize themselves to meet changing patterns of usage. Finally, to return to our discussion of a theory, we need a framework for problem description which encompasses not only the data and the logical relationships, but also such operational factors as volumes, elapsed time, reliability, etc.; very few existing languages, for instance, have such adverbs as "slowly" and "carefully."

Let us now turn to the second interpretation of the work group subject, adopt a more theoretical approach, and show how such a theory could, in fact, shed light on the problems of user communication.

At the outset, it was decided that we must deal only with a data base consisting of formatted data. Even though this data should be easier to handle than unstructured natural language, there still is no reasonably comprehensive theory of well-structured or formatted problems, especially where the user may be in a sense browsing through the data trying to define his problem.

A theory of problem definition should be as follows:

1. Independent of method of solution (i.e., mechanization). It should be possible to describe a problem before a decision is made as to whether the files should be random or sequential (granted, of course, that for some problems only one decision is possible).

2. Capable of describing specific processing when it is advantageous to do so. A user should be able to describe not only the problem but also as much of the solution as has been determined.
3. Able to describe the problem at any level, or at any stage in the solution-defining process, including those at which the problem information is incomplete or inconsistent. (Several methods exist which require all the "blanks" to be filled in before any work can proceed.)
4. Capable of useful mechanical manipulation (like algebraic notation) to aid in the analysis of a problem and its solution.

At this point, the majority of the work group members decided that a theory could be developed most aptly by discussing a grammar and syntax for data base problem definitions. Four elements are required to develop this theory. The theory (or grammar) can be started at the same place that the analyses of the problem was begun, i.e., by establishing the set of data elements required. This is, of course, a most complex task, but some of the items mentioned were:

- . Definition: The definition can be by narrative, by listing the elements, or by giving a rule for set membership. In another sense the set is also defined by those elements added to or deleted from it during processing.
- . Name and synonyms
- . Number of elements
- . Codes
- . Authority for maintenance
- . Security classification

The second item required is a subset of a data set; this subset may be defined in the same way that a set is defined, or that a collection of all the elements of a set which satisfy a certain condition is defined

The third element required to develop a theory for the data base problem is composed of the relationships among data elements. One of several possible ways of specifying these relationships is by various collections of sets of data values. It is tempting to think of these collections as files consisting of records; actually, however, that is a restricted and misleading view. An explanation of the differences (from a theoretical standpoint) is quite complex, and nearly impossible without drawing pictures. At any rate, we have one-to-one, one-to-many, and many-to-many relationships of various sorts. In addition, we do not make the apparently unnecessary distinction between entities and attributes, since they are both elements of certain sets.

The fourth element, ordered sets, can be either ordered operationally (i.e., by an implicit value as in a queue) or by the value of some component of the element. In a general problem description method, sorted sets enter, of course, only in defining output reports and not for processing purposes.

These four points define the part of the part of the theory that defines the data base.

In addition to the definition of the data base, of course, we also need operations for the accomplishment of user requirements and for the maintenance of our data base. There is little point in going into great detail here; operations are required to add or delete an element to or from one of a collection of sets, add a new set to the system, look up a value, and so on. It should be emphasized that even the simplest of these operations, when closely examined, turns out to be quite complex, and requires very careful definition. We can also think of a hierarchy of more and more complex operations, and, of course, of collections of operations (e.g., subroutines).

We have further to specify the input and output functions. As a beginning, we can distinguish three kinds of input:

1. Scheduled
2. Initiated by a user
3. Requested by the processing

The following three types of output are also commonly used:

1. Scheduled
2. Triggered by an input
3. Triggered by the occurrence of some state in the system (e.g., when a set becomes empty)

Also this part of the "grammar" presumably contains the operational considerations mentioned earlier, such as elapsed time.

Obviously what the work group produced, in the limited time available, is not a theory of problem definition, but only a few indications of some of the phenomena which a theory must encompass, and some ideas on the relationship between theory and practice in this enormously complex and difficult world of information processing problems.

## CRITERIA FOR EVALUATING DATA MANAGEMENT SYSTEMS

Chairman: Herbert R. Koller, Research & Development Division, U.S. Patent Office.  
SDC Associate Chairman: Joel M. Kibbee

This work group was concerned with a wide variety of problems that must be confronted to establish criteria for evaluation of data base systems.

Although information systems of this general character have received attention for over a decade, very little is known about how to evaluate them. As noted by Bryant,<sup>1\*</sup> scarcity of information may be attributed to a number of causes, among which are a lack of well-defined objectives for systems, a wide spectrum of both applications and users, uncertainties in developing meaningful measurement criteria, and uncertainty in the meaning of the concept of "relevance."

One method of developing performance and cost criteria was contributed early in the group's discussion; this method was the possibility that hardware manufacturers may well include compilers for data base system operation in their software packages. The question of standardizing this class of programming languages was raised; in general, this type of standardization was felt to be premature at the present time.

A system may be evaluated in terms of user satisfaction as well as in terms of economic considerations. For some purposes, it would be advantageous to treat these two areas separately, although the distinctions between them may, at some points, be vague. For other purposes it would be useful to have a single complex criterion which would provide an over-all evaluation of the system in all of its aspects. An example of the latter which was suggested by one member of the group was: to what extent and in what ways does the system effect changes in performance of the organization it serves?

Many specific criteria were suggested in the discussion. The wide gamut of possible applications of data base systems, their variety of sizes, the differences in users and their aspirations, the range of sophistication of equipment that may be used, and other variable factors, prohibit application of all possible criteria to all systems.

Standard criteria should be developed to which weights could be assigned to reflect the relative importance of the criteria for any particular application. This approach to systematic evaluation could afford a basis for evaluating a given system against some meaningful ideal, as well as providing a basis for comparative evaluation of different systems suggested for a single application.. A similar approach to the selection of data processing equipment by the Department of Defense has been suggested by Rosenthal.<sup>2</sup> A corollary approach to

---

\*Numbers in superscript refer to references presented at the end of this Work Group Report.



evaluation is represented by the development of so-called "benchmark" problems; these problems are calculated to exercise a variety of features of a system. Comparisons of systems operating on the same problem may have considerable value. Unfortunately, the work group did not have sufficient time to develop tools of these types; a serious effort to develop them should be attempted, preferably by pooling government and industry resources.

Another device, which could be used not only for evaluating and comparing systems, and which could also be used as an information disseminating means, would be a standardized language for describing data base systems. Provision should be made in the language for describing the capabilities, parameters, hardware requirements, and special features of the system.

The following criteria to evaluate particular measures of performance and aspects of data base systems were discussed:

- . the currency of information available through the system
- . the adequacy of retrieval
- . the simplicity of notation employed
- . whether the system is an on-line, batch, or job-shop system
- . documentation available on the system
- . flexibility in choice of output medium and format
- . the ease of integration with other systems
- . the flexibility in query procedure
- . the extent to which the user can pose queries relevant to his current needs
- . characteristics of the search strategy
- . the transferability of software used in the system
- . response time
- . the degree of interaction permitted between the user and the system
- . whether a demonstrable piece of hardware or software exists
- . open-endedness of the files
- . training requirements for users
- . checking features for data validity
- . ease of changing, altering, or extending the system or the data
- . the ease of entering new data
- . the direct, indirect, and negative costs
- . whether the system can treat programs as part of the data base
- . the presence of data processing capabilities
- . the relevance of answers retrieved by operation of the system.

Apparently, data base systems have much in common with document storage and retrieval systems which store a list of descriptors as a condensed surrogate for the full text of the document. A considerable body of literature exists that describes and evaluates document storage and retrieval systems; some of this literature is of interest to those who are concerned with evaluating data base systems. In this connection, the findings of a conference sponsored by the National Science Foundation in 1964<sup>3</sup> are pertinent. A comprehensive bibliography on this topic is being prepared.<sup>4</sup>

Some of the difficulties in establishing evaluative criteria stem from the difficulty in defining the user of the system, identifying his objectives, and determining how he interfaces with the system. Several classifications of criterion types which were suggested were based primarily upon the particular type of user being considered. One proposal, for example, suggests classification of a user in one of the following ways:

- . Data oriented--one who has need for information or who contributes data to the file, but who is naive with respect to the inner workings of the system.
- . System oriented--a designer, programmer and operator.
- . Interactive--one who literally poses queries directly to the hardware, via software, or, in the case of an on-line system, the console operator.
- . The ultimate user--one whose need to make a decision, for example, causes a query to be put to the system.
- . The manager--a person who must, in effect, evaluate the system in terms of whether or not his staff has effectively and efficiently been able to secure the best information to perform their proper job functions.

User-need satisfaction criteria would differ for each of these classes of persons.

A second proposal was made that evaluation criteria be classified as: performance/cost; user interaction; credibility; and social impact criteria. An alternative grouping was: performance criteria (including economic); user difficulty and training criteria; external circumstances; and interaction with society.

On the economic aspects of data base systems, it is important to recognize and evaluate all of the costs of designing the system, creating the files (a factor which is frequently grossly underestimated); maintaining and updating files; operation of the system in productive use; equipment costs, and other direct and indirect costs, some of which may be treated as the capital, depreciable expenses, and others as the daily costs of doing business. However, it is equally important, but considerably more difficult, to determine the value to the organization of the information obtained as the output of the system, or the cost of being unable to obtain information, and related "negative costs."

The following actions were encouraged by the work group:

1. The development of a systematic basis for purging files of obsolete and erroneous data.
2. The development of programming language and compilers for data base systems as parts of the software package developed for new hardware systems.

3. The further proliferation of new systems because of the many types of potential users and the many kinds of applications for data base systems. It is of interest to note a comment from one member of the group that proliferation of systems may continue by default in the absence of methods for standardizing and evaluating these systems.
4. Further research on data base systems without a requirement that the systems have immediately useful applications.

Several topics were suggested for further consideration and discussion:

1. The standardization of data base query and programming languages, and the development of a standardized description language for data base systems.
2. The development of specific, standard, well-defined criteria for evaluating data base systems, with suggestions for appropriate weighting of criteria for particular applications.
3. Further development of the methodology of evaluation, giving appropriate consideration to experimental design, adequate mathematical models of data base systems, methods of controlled experimentation, the application of appropriate statistical techniques, and a well-considered definition of "relevance."

#### REFERENCES

1. Edward C. Bryant, "Progress Toward Evaluation of Information Retrieval Systems," Proceedings of the ICIREPAT Conference, October 1964, Spartan Books (1965).
2. Solomon Rosenthal, "Analytical Technique for Automatic Data Processing Equipment Acquisition," Proceedings of the 1964 Spring Joint Computer Conference, April 1964, Spartan Books (1964).
3. "Summary of Study Conference on Evaluation of Document Searching Systems and Procedures--2 and 3 October 1964," Sponsored by the National Science Foundation, 10 February 1965.
4. Madeline B. Henderson, "Tentative Bibliography on Evaluation of Information Systems," Research Information Center and Advisory Service on Information Processing, National Bureau of Standards. (In press, scheduled to be published by the end of 1965.)

## RECORDING FOR ANALYSIS, COSTING, AND CONTROL

Chairman: Colonel Paul G. Galentine, Jr., Electronic Systems Division  
SDC Associate Chairman: George F. Weinwurm

As a point of departure it might be well to observe that the value of this work group discussion stems from two causes: the topic under consideration and the diversity of the participants. The group was composed of representatives from suppliers of computer systems (both hardware and/or software), users of such systems, government and industrial decision makers whose offices are charged with the responsibility of evaluating and passing upon proposed purchases, and management consultants who frequently act to advise their clients as to the desirability and cost-effectiveness of EDP installations. All were gathered to consider the common problem of measuring the cost-effectiveness of EDP and its applications.

While space does not permit the identification of each participant and his particular contribution, a number of broad areas of concern were evident throughout the two days of discussions, and are briefly summarized in this report.

The absence of generally meaningful ways of evaluating the economic performance of computer systems, about which each group of participants expressed a strong conviction, seems to be a problem that is common to nearly every part of the information processing field. To the extent that the work group constituted a representative and rather expert sample of opinion, one might conclude that broadly useful measures of cost and effectiveness for EDP remain to be developed and demonstrated.

The data processing manager's problems are especially urgent because of the apparent and seemingly obvious advantages of computerization. Whether these apparent advantages are real and lasting and are not contravened by such diffuse and obscure, but often critical, factors as ineffective utilization, insufficient experience with automatic data processing, inappropriate organization, etc., is difficult to establish. In the absence of clear evidence to the contrary, the more easily demonstrated virtues too often carry the day.

One of the problems that makes the measurement of EDP cost-effectiveness especially difficult is that the operation of the computer is necessarily intertwined with the flow of information throughout the using organization. Because of the special capabilities of EDP equipment, decisions as to proposed installations should be based on a review or systems analysis of the user's organizational structure, and on an appropriately designed information flow, in which EDP will have its proper part. Instead, EDP is often used simply to replace manual or semiautomated systems, performing identical functions albeit with greater apparent speed and fewer personnel.

One reason is that the virtue of EDP as opposed to an existing manual or semiautomated system is far easier to demonstrate than the degree to which the full potential of the proposed installation will be realized. Although cost-effectiveness decisions with respect to EDP tend to be made ad hoc more than by generally accepted comparative procedures, the fact remains that they are made continually in government and industry, and that rather reliable ways of evaluating the initial direct-conversion to EDP have been developed. Measuring from a cost-effectiveness viewpoint the extent to which a proposed installation makes the best use of EDP capabilities in a certain context is another matter. While such factors as equipment rental and site preparation are not difficult to determine, and are usually taken into account, other costs, such as the modification of files, forms, procedures, and organization are much more difficult to estimate reliably and are often overlooked or overly simplified. Certainly, the failure to consider the total impact of EDP, to make the necessary organizational changes and meaningfully reflect them in cost-effectiveness estimates, often lies behind the all-too-common complaint: "We now have all the clerks and the computer besides."

These problems are compounded by the rate of technical change in computer systems (both hardware and software), as well as the range of actual and potential applications. Such developments as time-sharing, procedure-oriented languages, and the relatively sophisticated types of man-machine interface equipment that are becoming increasingly evident will undoubtedly result in substantial changes to information processing as we now know it (all too imperfectly). Just how pervasive these changes will prove to be is difficult to predict. More certain is that other equally significant innovations will follow. The managerial lessons learned by other fields, in far less dynamic days, suggest that the evolutionary process by which ad hoc rules of thumb are transformed into generally accepted and applicable standards and techniques takes a long time, far longer than we can afford, or would be willing to wait. Also, the rate of evolution of new technology is outstripping the rate of evolution of management techniques to cope with the existing technology.

Moreover, considerations of cost-effectiveness are only the final link in a chain of understanding that presupposes measures of both relative cost and effectiveness, comparably structured data from which these can rationally be derived, and generally agreed upon concepts in terms of which the data can be collected from the start.

In retrospect, the underlying theme of the work group discussions was that the foundations upon which cost-effectiveness measures must be based--the common concepts, the definitions, the data, the analysis--are missing.

1 December 1965

5-29  
(Last page)

TM-2624/100/00

Proper exploitation of new technology requires a solid foundation. Our relative inability to learn from past experiences due to the lack of meaningful cost data is particularly disturbing. There is an urgent need to establish meaningful cost recording techniques, followed by extensive analyses of the resulting data. These steps seem fundamental to the formulation of improved EDP cost-effectiveness and management procedures.

It seems clear that members of the information processing profession will have to develop better ways to measure and control cost-effectiveness, and soon, if these are not to be imposed by others. The recent "Clewlow Committee" report<sup>1</sup> accurately reflects the sense of urgency that is felt at higher levels of the government that "something" must be done to assure effective management control of EDP installations, present and proposed. The extent to which these management techniques are developed in an orderly and defensible fashion should be a matter of immediate and lasting concern in government and industry.

---

<sup>1</sup>Committee on Government Operations, U. S. Senate, "Report to the President on the Management of Automatic Data Processing in the Federal Government," March, 1965. This report was prepared for the Bureau of the Budget by a Project Staff including Carl W. Clewlow.